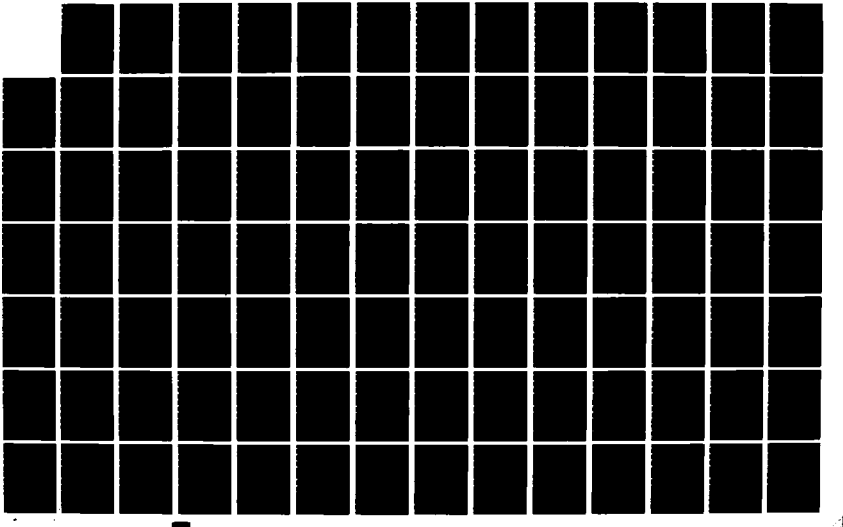
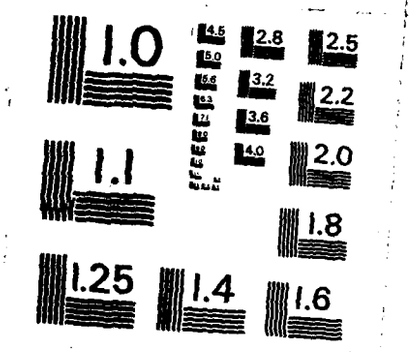
AD-A166 846 A HIGH-SPEED ASYNCHRONOUS COMMUNICATION TECHNIQUE FOR 1/2
MOS (METAL-OXIDE-SE (U) MASSACHUSETTS INST OF TECH
CAMBRIDGE DEPT OF ELECTRICAL ENGIN P D BASSETT
UNCLASSIFIED DEC 85 VLSI-MEMO-85-283 N00014-80-C-0622 F/G 20/12 NL





DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

CAMBRIDGE, MASSACHUSETTS 02139

VLSI Memo No. 85-283 December 1985

A High-Speed Asynchronous Communication Technique for MOS VLSI Systems*

Paul D. Bassett**

ABSTRACT

As MOS technologies advance, the relative differences between on-chip and off-chip delays increase. Drivers and receivers can be designed which allow high bit rate communications (>100 Mbits/sec) between MOS chips at the expense of increased latency. Designing synchronous systems which couple a high clock frequency with large and variable delays is difficult and expensive due to the complexity of insuring that no delays violate the constraints imposed by synchronous operation.

A circuit-based technique for automatically adjusting signal delays in an MOS system has been developed. The Dynamic Delay Adjustment (DDA) technique provides reliable high speed communications directly between MOS chips independent of the delay between the chips. The amount of phase jitter immunity provided by the synchronizer can be traded off against circuit complexity; the signal delays are adjusted continuously to track temperature induced delay variations. A 3 micron CMOS DDA synchronizer has been fabricated to confirm the validity of the DDA approach; test results will be presented.

*Submitted to the Department of Electrical Engineering and Computer Science on May 14, 1985 in partial fulfillment of the requirements for the degree of Master of Science. This work was sponsored in part by an RCA Fellowship and in part by the Defense Advanced Research Projects Agency under Contract number N00014-80-C-0622.

**Current address: Bolt, Beranek and Newman, Room 6/501, 10 Molton St., Cambridge, MA 02238; (617) 497-2471.

A High-Speed Asynchronous Communication Technique for MOS Systems

by

Paul D. Bassett

B.S. Electrical Engineering, Texas A&M University (1982)

Submitted in partial fulfillment
of the requirements for the degrees of

Master of Science
and
Electrical Engineer
in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology
May 1985

Signature of Author Paul D Bassett
Department of Electrical Engineering and Computer Science
May 14, 1985

Certified by
Lance A. Glasser
Thesis Supervisor

Accepted by
Arthur C. Smith
Chairman, Department Committee on Graduate Students

A High Speed Asynchronous Communication
Technique for MOS VLSI Systems

by

Paul D. Bassett

Submitted to the

Department of Electrical Engineering and Computer Science

on May 14, 1985 in partial fulfillment of the requirements

for the degree of Master of Science.

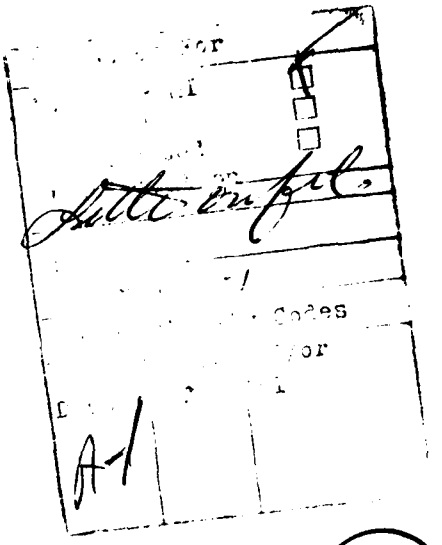
Abstract

As MOS technologies advance, the relative differences between on-chip and off-chip delays increase. Drivers and receivers can be designed which allow high bit rate communications (> 100 Mbits/sec) between MOS chips at the expense of increased latency. Designing synchronous systems which couple a high clock frequency with large and variable delays is difficult and expensive due to the complexity of insuring that no delays violate the constraints imposed by synchronous operation.

A circuit-based technique for automatically adjusting signal delays in an MOS system has been developed. The Dynamic Delay Adjustment (DDA) technique provides reliable high speed communications directly between MOS chips independent of the delay between the chips. The amount of phase jitter immunity provided by the synchronizer can be traded off against circuit complexity; the signal delays are adjusted continuously to track temperature induced delay variations. A 3 micron CMOS DDA synchronizer has been fabricated to confirm the validity of the DDA approach; test results will be presented.

Thesis Supervisor: Lance A. Glasser

Title: Assistant Professor of Electrical Engineering and Computer Science



Acknowledgments

I would like to express my sincere thanks to:

Lance Glasser who has taught me almost everything I know about VLSI circuits, contributed greatly to the ideas in this thesis and has been very supportive in my 3 years at MIT.

Bob Armstrong, Steve McCormick and Guy Fedorkow for designing, writing and maintaining the CAD tools without which all of my circuit design efforts would have been very much more difficult.

Scott Wills, John Harris, Cyrus Bamji, Eric Tenenbaum and all my other office mates and neighbors for providing many hours of interesting discussions and timely distractions.

Randy Rettberg, Will Crowther, John Goodhue and the rest of the Multiprocessor Hardware Group at Bolt, Beranek and Newman for contributing important ideas and helpful suggestions to this thesis and for greatly expanding my understanding and knowledge of multiprocessors and computers in general.

My parents and all of my family who have always been very supportive and encouraging in all of my efforts — academic and personal.

This work was sponsored in part by an RCA Fellowship and in part by the Defense Advanced Research Projects Agency under Contract No. N00014-80-C-0622.

This thesis is dedicated to Jane who has patiently waited through the longest 2 years of an MIT graduate student's life — the *last* semester.

Contents

List of Figures

Chapter 1

Introduction

In the past 15 years or so, Metal-Oxide-Semiconductor (MOS) technologies have had a major impact on the computer field. The effect of MOS technologies has been based on the rapidly increasing scale of integration acheiveable; the number of devices which can be placed on a single chip has been doubling every few years and will probably continue to do so for several more years. MOS technologies also have the ability to store charge dynamically for periods of time long relative to the device switching times. The combination of circuit density and dynamic storage have made possible such developments as the 1 transistor dynamic RAM, the EPROM and the single-chip microprocessor. With the exception the of the dynamic RAM which has made possible the multi-megabyte main memories of large computer systems, the impact of MOS technological developments has been felt mainly in the lower performance end of the computer spectrum.

MOS has remained a low end logic technology because bipolar technologies have always had a significant speed advantage over MOS. All of the present generation supercomputers computers are based on high speed bipolar Emitter-Coupled-Logic (ECL) families. These technologies are very fast but consume large amounts of power; this limits the amount of circuitry which can be placed on a single chip to orders of magnitude less than is achievable in MOS. Machines constructed using ECL components have tended to be very fast but also very expensive to design, construct, operate

and maintain.

As the minimum dimensions drop into the submicron range, many of the tradeoffs are changing. As technologies scale, dimensions tend to scale roughly together in all three dimensions. Bipolar devices are constructed vertically and the switching speed of the bipolar devices depends on the width of the base region. MOS devices are constructed horizontally and the switching speed of MOS devices depends on the current drive capabilities of the devices and the capacitive load presented by the gates of the devices. When all dimensions scale down by a factor of $1/\tau$, the switching speed of the bipolar devices will increase by a factor of τ due to the decrease in the base width; the speed of the MOS devices will increase by a factor or τ^2 due to increases in the currents by a factor of τ and decreases in the capacitive loads by the same factor. As the speed advantage of ECL decreases, the density and power advantages of MOS technologies become more important in the decision between the technologies. With the current industry wide switch from NMOS technologies to Complementary MOS (CMOS) technologies, the power advantage of MOS has increased even further.

As MOS technologies move into the higher speed applications, new concerns are becoming important that have not been addressed by MOS designers previously. Specifically, in past MOS based systems the clock frequency and data rates have typically been so slow that inter-chip communications have not been a serious problem. As frequencies climb into the 50—100 MHz range, the delay associated with sending data from one chip can easily be equal to or greater than the clock period.

ECL machine designers have been addressing this problem for years but the change in technologies has added new twists to the problem, as well as introducing the possibility of using new approaches to solving the problem.

1.1 Synopsis

Chapter 2 will discuss the constraints which the designer of a high speed computer must follow in order to insure the computer operates properly.

Particular attention is given to how characteristics of the MOS technologies and assumptions regarding the clocking strategy impact the constraints. Chapter 3 will introduce a circuit based synchronization technique known as Dynamic Delay Adjustment and discuss an implementation of the technique in a CMOS technology. The implementation has been fabricated and tested and the results are presented and discussed. Appendix A presents an analysis of the bistable latches in order to predict the probability of a synchronization failure occurring in the DDA synchronizer.

Chapter 2

Delay Constraints in High-Speed Synchronous VLSI Systems

This chapter will discuss delay constraints in high speed synchronous systems based on MOS VLSI chips. Specifically, the constraints involved with transferring data between chips in such a system will be of the most interest. The architecture of the system will be assumed to be highly pipelined using a two phase non-overlapping clock to control the flow of data between pipeline stages.

Figure 2.1 illustrates how communication between chips will be modeled. The XMTR chip is sending data to the RCVR chip. The output of the XMTR is simply a latch which is clocked by ϕ_2 followed by a buffer; the input to the RCVR consists of another latch, clocked by ϕ_1, which is possibly preceded by an amplifier. The line connecting the two chips is shown as terminated transmission line; the line could also have been modeled as a simple capacitor.

The constraints this type of system imposes on the designer will be the subject of this chapter. Although the synchronous model used here is the most commonly used timing scheme, this scheme is used only to provide a framework for the discussions in the chapter. Other assumptions and timing

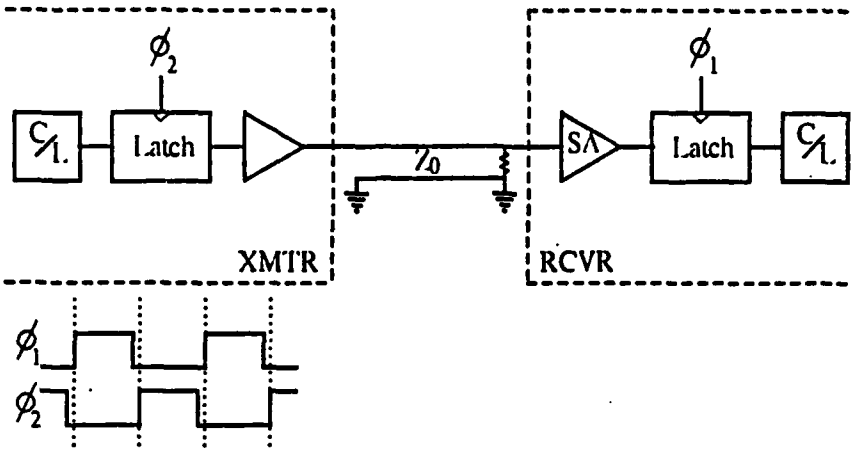
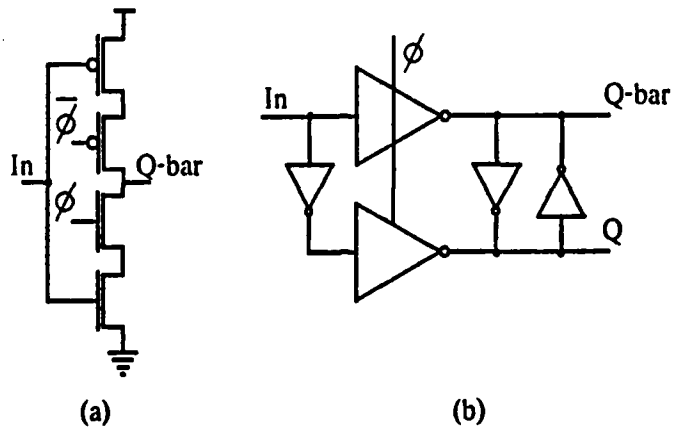


Figure 2.1: Modeling Inter-Chip Communication

models could just as easily be used; in most cases the problems discussed and conclusions drawn in this chapter would still be valid.

2.1 CMOS Latches Under Marginal Switching Conditions

Figure 2.2(a) illustrates one of the simplest types of dynamic latches possible in CMOS a technology, a series combination of 2 p-type and 2 n-type transistors; a slightly more complicated static latch is shown in Figure 2.2(b), which uses 2 of the dynamic latches and 2 cross coupled inverters. Both of the latches operate in a manner which is fundamentally different from the *edge triggered* latches commonly used in TTL and ECL machines; these latches are more accurately described as either *level-triggered*. Edge-triggered latches are designed so that the outputs only change during a very short time, known as the *hold* time or t_H, immediately following the rising edge of the clock. For latches like the 2 CMOS latches shown, the outputs will respond to transitions on the data input with some small delay as long as ϕ is HIGH. As will be seen in forthcoming sections, this behavior of the CMOS latches has a big effect on the types of clocking schemes which can



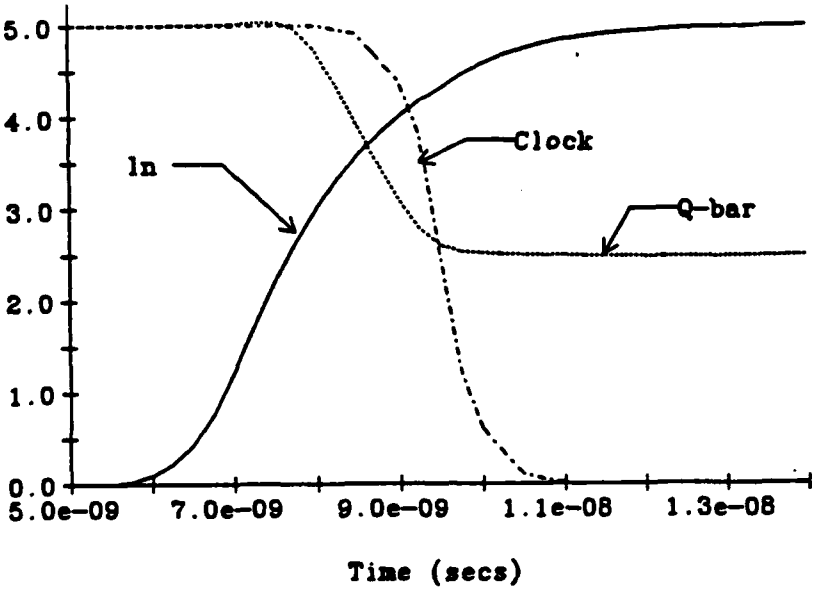
(a) (b)

Figure 2.2: CMOS Latches. (a) A Simple Dynamic Latch (b) A Bistable Latch

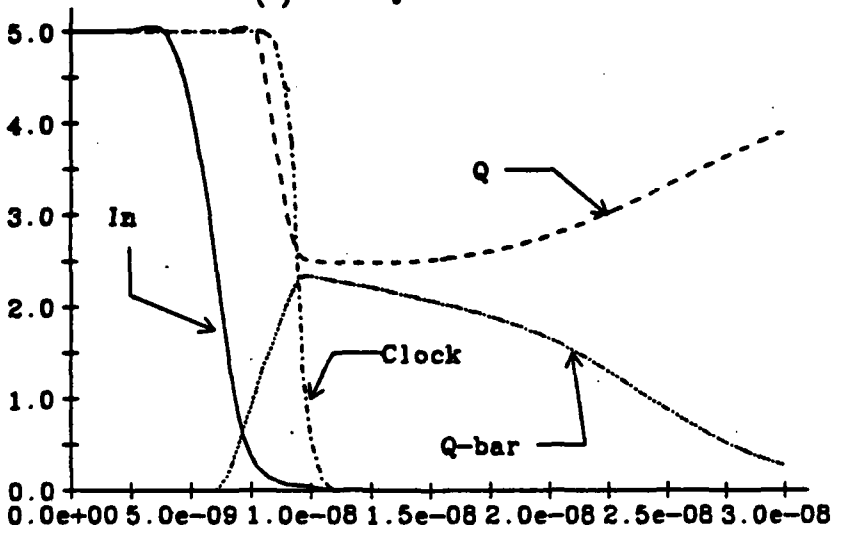
be used in MOS based systems.

Both types of latches have relatively short delays. For the dynamic latch the delay will be slightly longer than a typical inverter delay due to the two devices in series in both the pullup and the pulldown paths; for the static latch the delay will depend on the relative sizes of the devices in the inverters and in the latches; in order for the state of the latch to switch, the latches devices must be much greater than the devices in the inverters. Under normal switching conditions, data transitions occur far enough before the falling edge of the clock that the outputs have settled to logically valid levels. When data transitions happen very close to the falling clock edge, the outputs may not have sufficient time to switch completely. Figure 2.3 show the responses of these two latches to data transitions which fall very close to the clock edge. Both latches are assumed to drive some moderate load capacitance consisting of other logic gates.

The dynamic latch's output fails to switch completely before the clock signal turns off the two middle devices in the stack. Once these devices are off, the output impedance of the latch becomes very high; this high impedance prevents the capacitance loading the output from either accumulating or losing charge. The output of the latch will stay at the intermediate



(a) The Dynamic Latch



(b) The Bistable Latch

Figure 2.3: CMOS Latches Under Marginal Switching Conditions

value until ϕ goes HIGH again on the next cycle.

The situation is different for the bistable latch; the clock signal going LOW before the outputs are stable stops the output transitions just before the two output voltages are equal. However, the feedback present in the latch amplifies the small differential voltage and forces the outputs back to legal logical values. Appendix A will consider the bistable latch in much greater detail.

The job of the system designer is to insure that failures such as these do not occur very often. In order to see just how low the probability of failure must be in order to insure reliable system operation, consider the following example of a high performance multi-processor. Assume there are 10,000 processors each of which has a single, serial port which runs at 100MHz; the switch network required to interconnect this many processors would probably require around 5 stages of switching elements with each stage having on the order of 10,000 inputs which must be latched on every cycle. Assuming the computer operates 24 hours a day for 250 days a year[1]; in order to have one latch failure a year on the average, the probability of one of the latches failing on any particular cycle would have to be less than 10^{-20}.

2.2 Specification of MOS Logic Families

The previous section illustrated one type of behavior that can result from improperly latching signals in MOS systems. In order to avoid such situations, designers must be able to predict the performance of the MOS components before they are fabricated. This section will discuss how the specifications required by a digital system can be mapped onto a technology like CMOS. The behavior of the most interest here is the dynamic behavior so only a brief discussion of the static characterizations will be given.

[1]Leaving almost 1/3 of the year unused.

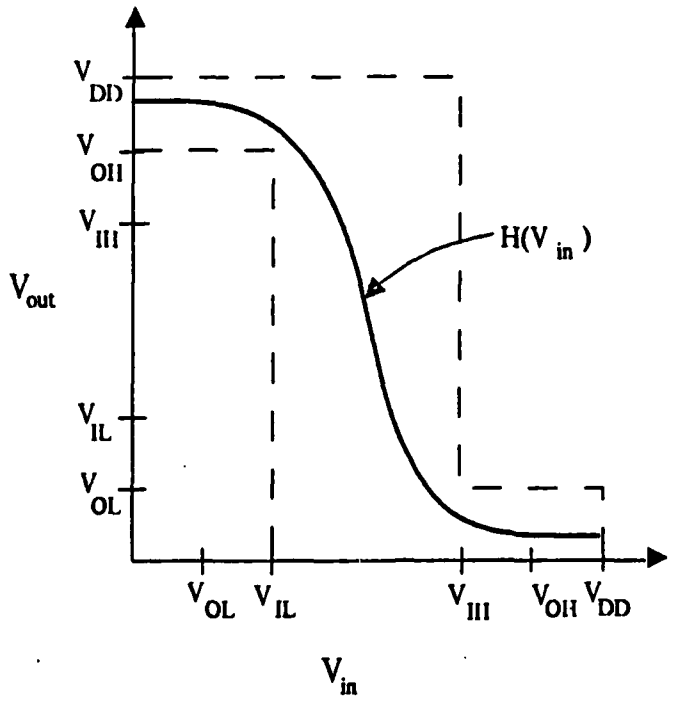


Figure 2.4: An Inverter's Static Transfer Characteristics and the Corresponding Logic Families Level Specifications

2.2.1 Static Specification of MOS Logic Families

Figure 2.4 illustrates the static electrical specifications for a hypothetical MOS logic family; the specifications consist of separate HIGH and LOW voltage levels for inputs, V_{IH}, V_{IL}, and outputs, V_{OH} and V_{OL}[12]. Any logic gate in this family must obey the requirement that if all of the gate's inputs are either above V_{IH} or below V_{IL} then the gate's output must be either above V_{OH} or below V_{OL}. The shaded portion of the figure shows the portion of the V_{in} vs. V_{out} plane in which valid logic signals must fall.

Every logic gate will have a unique transfer function which characterizes the output voltage as a function of the input voltage(s). Figure 2.4 also shows the transfer function for an inverter whose static transfer function

$v_{out} = H_S(v_{in})$ meets the requirements of the logic family; that is:

$$H_S(v_{IL}) > v_{OH} \tag{2.1}$$

and

$$H_S(v_{IH}) < v_{OL} \tag{2.2}$$

One point that is immediately obvious is the fact that this emulation of a binary system is basically a failure in the sense that there are states in the technology which are logically undefined; this is not the case for a truly binary system. It is exactly this failure which causes the kinds of problems seen in Figure 2.3; in a truly binary system, latching an illegal output state would be impossible. Unfortunately it is impossible to build a truly binary logic system output of the continuous mediums available.

2.2.2 Dynamic Specification of MOS Logic Families

Just as the logic levels were directly related to the static electrical characteristics of the logic gates comprising a family of digital logic, it should be possible to characterize the dynamic performance of the same gates. An important concept in the characterization presented here will the use of *characteristic waveforms*; assuming that all signals of interest have some typical waveshape will simplify the definition of the delay between signals. A standard method for determining the delay between the data and control inputs to a gate and the resulting data outputs is necessary for characterizing the dynamic performance of a MOS logic family.

Characteristic Waveforms

In MOS technologies logic gates drive loads which are almost purely capacitive. The major components of these loads are the gate and Miller capacitances of the MOS transistors being driven and the parasitic capacitances due to the diffusion, polysilicon and metal used to interconnect the devices. When long lines of polysilicon or metal interconnect must be driven, there

can also be significant resistive and possibly even inductive components in the load but these cases are not typical.

The nonlinear nature of the devices and the gate capacitances make it difficult to generate closed form descriptions of the waveforms. However, by making a few simplifying assumptions it is possible to accurately predict the responses of MOS logic gates to simple inputs like steps and ramps [18] [14]. The exact form of the predicted responses will depend on the assumptions made and the definition of *delay* used. In any case however, the delay will be almost directly proportional to the electron and hole mobilities, the power supply voltage and the fanout ratio of the circuit. Other important parameters are the transistor thresholds and the ratio of the n-type transistor widths to the p-type transistor widths [18].

Deriving expressions for the response of single gates to simple input waveforms is helpful but the response of several gates cascaded together is much more interesting and useful to the circuit designer. It is more difficult to express the response of cascaded gates in any analytical form, but not impossible [14]. Figure 2.5 shows the response of a string of inverters to a step input. The device sizes have been chosen to have the fanout of every stage approximately equal.

It can be seen that after only a couple of stages the waveforms begin looking almost identical. This is a common characteristic of MOS circuits and makes it possible to discuss delays in terms of *characteristic waveforms* [2] rather than trying to characterize signals in terms of steps and ramps. The delay between two edges is defined as the time between the $V_{dd}/2$ points on each edge. This definition is not dependent on rise and fall times or even whether the edges are rising or falling.

This definition of delay between signals can be extended to define the phase margin, t_{PM}, of a clocked latch to be the delay between data and clock inputs to the latch. The next section will examine the relationship between t_{PM} for a latch and the resulting performance of the latch.

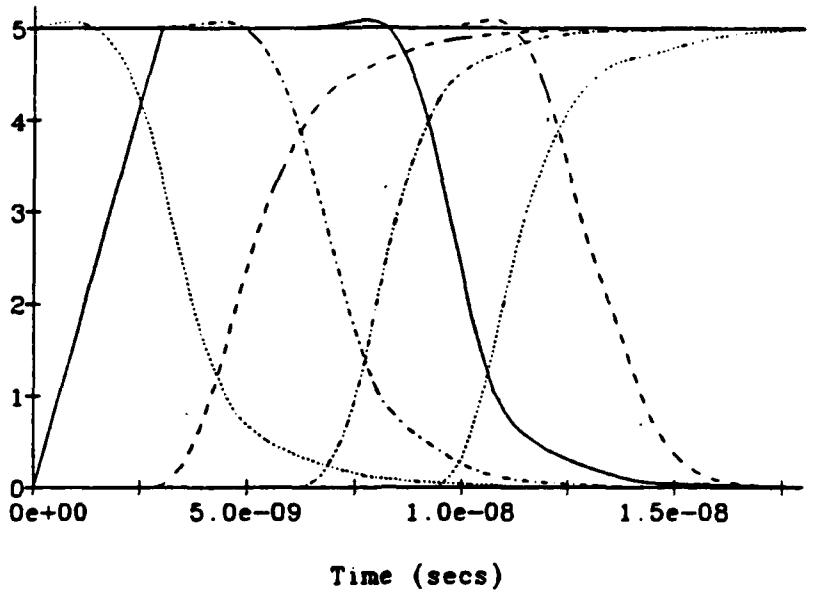


Figure 2.5: The Step Response of a String of CMOS Inverters

Dynamic Transfer Functions of CMOS Latches

Section 2.1 illustrated the response of both dynamic and static CMOS latches to nearly simultaneous data and clock transitions. This section will consider in more detail how the response of the dynamic latch depends on the phase margin of the data input with respect to the clock. Similar characteristics could be studied for the static latch as well. Appendix A analyzes the static latch in a slightly different manner; that analysis will be used to extend this section's observations to static latches in a qualitative sense.

Figure 2.3(a) showed the response of a dynamic CMOS latch to a data transition just before the falling clock edge. For this particular arrangement of data and clock edges, the output is latched just above $V_{dd}/2$ and does not change until the clock goes HIGH again. Had the data transition occurred one nanosecond earlier, the output would have switched completely; a one nanosecond later transition would not have caused any movement of the output until the next clock cycle.

By simulating the latch for data transition times ranging over the 2ns interval just mentioned, the voltage at which the output is latched can be

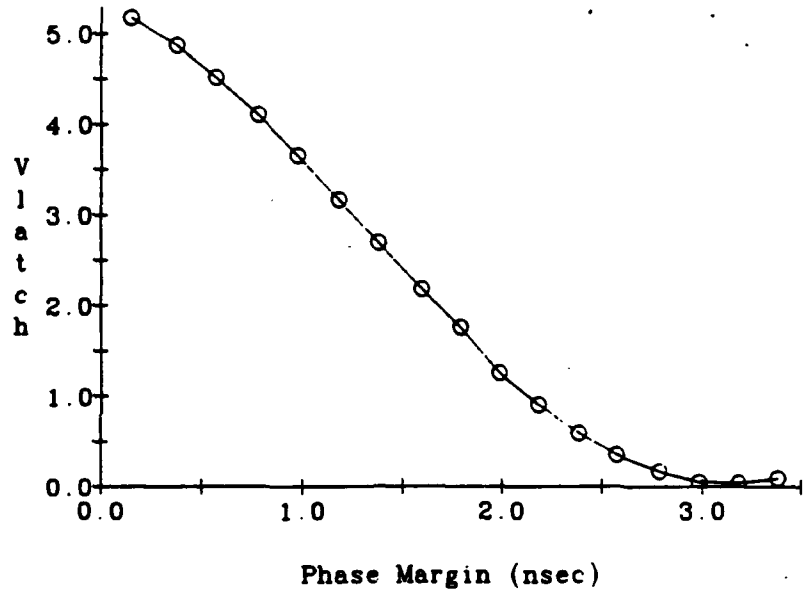


Figure 2.6: V_{latch} vs. Phase Margin for the Dynamic CMOS Latch

determined as a function of the phase margin of the data signal. Figure 2.6 shows the results of such a series of simulations; the figure plots the latched output voltage, V_{latch}, versus the phase margin, t_{PM}.

This plot defines the dynamic transfer function for the latch, $H_D(t_{PM})$; this transfer function is similar to the more familiar static one, $H_S(V_{in})$, shown in Figure 2.4 in that it provides a one-to-one mapping of an input characteristic to an output voltage. The static mapping was used to generate the specifications for the static logic level definitions; this dynamic transfer function will be used to define delay constraint specifications, i.e. setup and hold times. Section 2.5 will uses both sets of specifications to discuss how the relative noise immunity of a latch relates to the specifications.

There is some small window of phase margins which will result in a voltage being latched which violates the static logic voltage level specified for the technology. Due to the definition of delay, one of the boundaries of this window may be a negative phase margin, i.e. the data transition occurs slightly after the clock transition. By convention, this boundary will be called the hold time of the latch, t_H. The positive boundary of the window

will be called the setup time of the latch, t_S, also by convention. The setup and hold times must obey the following rules in order to insure the latch does not violate the static logic level specifications[2]:

$$V_{latch}(t_S) = H_D(t_S) > V_{OH} \tag{2.3}$$

and

$$V_{latch}(t_H) = H_D(t_H) < V_{OL}. \tag{2.4}$$

As long as the designer insures that the data and clock transitions always obey these setup and hold time specifications, the latch will never latch a logically undefined voltage.

Appendix A analyzes the behavior of the bistable latch in a slightly different manner; the conclusions drawn in the Appendix will be used to extend the discussion of this Section to bistable latches. The first conclusion the Appendix draws is that any non-zero initial differential voltage between the static latch's outputs will eventually be amplified until the outputs reach logically defined states. Secondly, if the phase margin is completely arbitrary, then for any amount of time the latch is given to settle there is a probability that the metastable state will persist longer than the settling time; the probability decays exponentially with increasing settling time.

Assuming that the system is using 2-phase non-overlapping clocks and one phase is clocking the input to the static latch while the other phase is sampling the outputs, the settling time of the latch would be $T_C/2$ seconds, where T_C is the clock period. Simulations could once again be used to generate an $H_D(t_0)$ transfer function for the static latch by taking V_{latch} to be one of the output voltages $T_C/2$ seconds after the first clock edge. Such a function would look similar to Figure 2.6 in most respects but the switching portion of the curve would be much sharper resulting in a much smaller difference between t_S and t_H for the static latch than for the dynamic latch. In situations where the phase margin of the input can not be well controlled, the static latch provides much lower failure probabilities than the dynamic latch.

[2]Assuming a rising output transition.

2.3 The Synchronous Operation Model

The previous section defined setup and hold time specifications for clocked latches which, when obeyed, insure that the latches' outputs will always logically valid. How the designer approaches insuring that these specifications are not violated depends on how the operation of the computer is being modeled. The most commonly used operating model is the Synchronous Model; the most basic assumption of the synchronous model is that all clock inputs transition simultaneously everywhere in the machine.

In a highly pipelined machine, all signals in the machine propagate between latches; and presumably at some point the phase margin of the signal to one of the clock signals is known. For instance, if the input to a latch is known to only change when the latches clock is LOW, the the output of the latch would always have a fixed phase margin with respect to the clock. By assuming skewless clock signals, the phase margin of any signal can be found by calculating the delays the signal experiences between the output of a latch at which the phase margin is known to the input of the latch at which the phase margin is needed.

The setup and hold characteristics of the latches limit the delays which are allowable on any particular path. These limits can be expressed as bounds the signal delays d_i which can be expressed as the sum of a logic delay d_{iL} and a wire delay d_{iW}. The clock frequency T_C and the setup and hold times will combine to provide the designer with bounds on legal values of d_i.

A simple synchronous model assumes the minimum delay of a stage of logic is greater than the required hold time but would require all delays be less than one clock period minus the setup time:

$$t_H \leq d_i = d_{iL} + d_{iW} \leq T_C - t_S \tag{2.5}$$

Another way of interpreting this constraint is that the clock period must be longer than the longest delay in the entire machine.

A more general synchronous model relaxes this single clock cycle require-

ment. Delays may be greater than T_C but the setup and hold requirements must still be followed. Equations 2.5 may be generalized by adding a multiple of T_C to each of the boundaries. In the following constraint k is an integer greater than or equal to 0:

$$kT_C + t_H \leq d_i \leq (k+1)T_C - t_S \tag{2.6}$$

Given a well understood logic technology determining logic delays may not involve much more than counting the number of stages of logic. Determining the wire delays requires a very thorough understanding of the physical organization of the final machine. The length of the longest wire delays with respect to the logic delays determines whether the two sided constraints are necessary.

If the one sided constraints are used, the clock period will be specified by the longest total delay; since the wire delays will vary from path to path, the designer has some flexibility to include more logic in those paths which happen to have shorter wires. However, the architecture of the computer will specify when operations can be performed, limiting this flexibility. Any portion of the clock cycle which can not be effectively used will be wasted and will lower the efficiency of the machine. The two sided constraints may make it possible for the designer to increase the fraction of the paths which effectively use the entire clock cycle; however, this increase in efficiency comes at a large expense in design complexity.

Clock Skew and Other Overheads in Synchronous Systems

One of the major assumptions behind the delay constraints 2.5 and 2.6 was that clock transitions occur simultaneously in all parts of the computer. In order to make this assumption hold, the clock distribution system must be carefully designed to insure each path from the clock source to each chip or group of chips has the same delay.

Matching the wire lengths along the clock distribution paths is feasible. The main source of variation in clock delays will be in the level converters

and buffers which will generally be needed to provide clock signals with sufficient drive at the chips. These converters and buffers may be placed on each individual chips or they may be placed externally and shared among a group of chips. In either case, process variations, thermal fluctuations and load variations will all be sources of delay mismatch leading to skewing of the clock transitions from chip to chip.

Clock skew due to controllable characteristics like wire lengths can be handled by the designer at the expense of further complexity in the delay constraints. Skew arising from thermal and process fluctuations is unpredictable and therefore the designer must estimate the worst case skew and allow extra time in the clock period to allow for the skew. This is pure overhead which lengthens the clock period without adding extra performance.

2.4 Delays: Types and Scaling

In order to prevent latch failures from plaguing a computer system, the designer must insure that no delay paths violate the setup and hold requirements of the latches used to latch the data inputs. In a VLSI system, this can involve working with both on-chip and off-chip delays and signals. This thesis is mainly concerned with inter-chip communication so the discussion here will concern primarily delays involving inter-chip signals. Section 2.4.1 will touch briefly on the use of equipotential regions to help simplify the delay constraints for on-chip signals.

This section will treat delays as if the actual delay values can be predicted accurately at design time. Due to manufacturing variations the actual delays present after the system is built may vary from the predicted delays; the designer must make the design robust enough to allow for the variations. The sources of delay variations and the effects of the variations on systems will be discussed along with other noise issues in Section 2.5.

2.4.1 Equipotential Regions

In MOS VLSI circuits, most of the on-chip wires used to interconnect devices and gates are short and can be accurately modeled as purely capacitive loads. However, on most chips there will be long lines of metal or polysilicon needed to connect distant portions of the chip together. These long lines may have a significant resistive component which will cause the line to appear more as a distributed RC line; signals tend to diffuse down such a line. There can be a noticeable skew between signals at opposite ends of the line. The circuit designer must therefore treat long lines differently when trying to account for delays in the circuit.

Whether a particular wire can be treated as a lumped capacitive load or not depends on how significant the delay associated with the wire is compared to the intrinsic logic delays in the associated circuitry. This classification has lead to the practise of dividing VLSI chips and systems into smaller units called *equipotential regions.* These regions are chosen small enough that wires contained within a region can be treated as lumped capacitances rather than diffusion lines. This simplifies the delay constraints within the region.

How large or small an equipotential region should be is open for debate. Charles Seitz uses the definition that *the maximum size for an equipotential region is that it be small enough that the potential on any wire within this area will equalize in less than* τ where τ is the delay of one inverter driving another inverter of the same size[30].

How much of a chip can be considered to be an equipotential region would be important to the chip designer, but for the system designer, the important point is that signals which pass between two chips will almost always cross the boundaries of an equipotential region. Delays on every wire between chips may have to be considered to insure proper operation.

2.4.2 Types of Delays

The main sources of delay which affect inter-chip communications in high

speed systems are:

- Driving large off-chip loads,
- Sensing small voltage swings,
- Transmission line delays.

Driving Off-Chip Loads

Driving off-chip loads can lead to significant delays due to the disparity between on-chip and off-chip dimensions. There are two ways in which off-chip loads can be modeled when designing buffer circuits, either as a pure capacitive load or as a transmission line with some characteristic impedance.

The strategies needed to drive each kind of load are slightly different but result in similar circuitry and delays. Figure 2.7 show buffers designed to drive capacitive loads and transmission lines including the parasitic capacitances and inductances which are due to the packaging technology used. The parasitics consist mainly of an *on-chip* capacitance, an inductor due to the bonding wire and package leads and an off-chip capacitance due to the package and pc-board interconnect. The capacitive load simply adds a large capacitor in parallel with the off-chip capacitance while the transmission line adds a transmission line in parallel with the off-chip capacitor.

The device sizes used in the final stage of the buffer will be determined by either the value of the load capacitance or the transmission line's impedance. In MOS technologies, the delay of any gate is roughly proportional to the fanout the gate must drive; a fanout of n will result in a gate which is n times slower than a gate with a fanout of 1. In order to have a fast rise time, the fanout seen by any of the inverters must be fairly small; the last inverter for the capacitive load will have to be very large. For the transmission line buffer, the last pulldown transistor must have a turned on resistance much less than the impedance of the transmission line, on the order of tens of Ohms or less. The large drive inverter and pulldown must in turn be driven quickly which requires transistors only a few times smaller than the

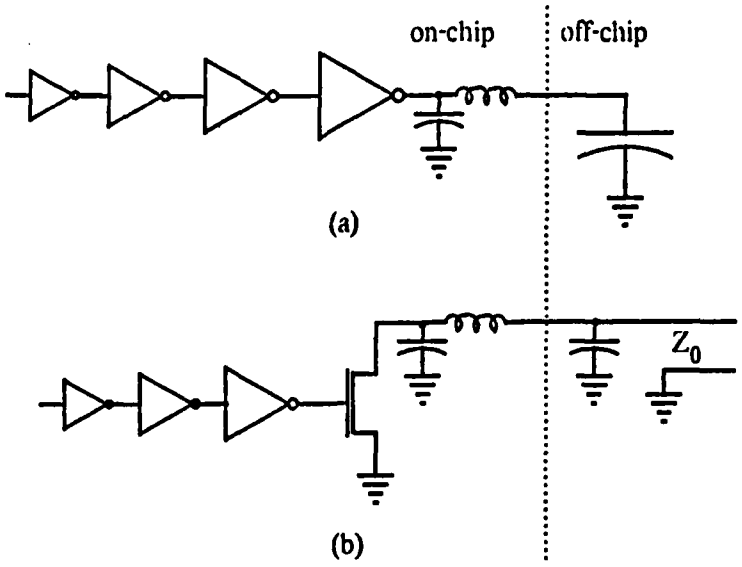


Figure 2.7: MOS Buffers Designed to Drive Off-Chip Loads (a) Capacitive Loads, (b) Transmission Line Loads

final drive devices. This reverse scaling continues until the resulting devices can be driven quickly by normal size devices; typically this type of scaling requires around 4 stages of inverters.

This buffering technique results in a buffer which has very fast rise times but a significant delay due to the latency required for the signals to propagate through the 4 or so stages of inverters. It can be shown that for capacitive loads the minimum delay required to drive a large load, C_L, is proportional to $\ln Y$ where Y is the ratio of C_L to the gate capacitance of a minimum size inverter[24]; a similar result should hold for driving transmission lines loads also. Because on-chip loads are scaling down much faster than off-chip loads, Y is increasing. As device sizes scale down, the minimum *absolute* delay required to drive off-chip loads is decreasing but the *relative* delay is increasing.

The latency does not have to be the deciding factor in setting the frequency at which data is sent through the pad however; the input to the buffer can be clocked as quickly as the delay through a single stage of the buffer

will allow. If the input is changed while there is still a transition propagating through the buffer, the buffer will act like a delay line and both transitions will appear at the output with approximately the same phase relationship as they had at the input to the buffer. This corresponds to choosing the two-sided delay constraints.

Sensing Small Voltage Swings

A power dissipation-delay tradeoff is required when choosing the voltage swings for off-chip signals. Provided the signals have short rise times, the power required to drive the off-chip capacitive loads is almost all ac power; as such the power is proportional to CV_S^2, where V_S is the maximum signal swing. The power required to drive the transmission lines is mainly due to the current required to pulldown the terminating resistor; this introduces a V_S^2/Z_o component to the power. In both cases, the power increases proportional to the square of the voltage swing of the interchip signals; this gives a strong incentive for lowering the signal swing. Due to gain-bandwidth limitations inherent in active devices, lower signal swings will require longer to sense. Luckily, due to the V_S^2 scaling of the power dissipation, large savings can be achieved without seriously impacting the clock frequency.

Transmission Line Delays

The other major component of inter-chip delay is the delay due to the pc-board traces and inter-board wires. Treating wires and traces of more than a foot or so in length as capacitive loads is probably not very feasible; in order to keep buffer sizes and delays reasonable and independent of the actual loads, transmission line loads are necessary. Signals propagate through these transmission lines at far less than speed-of-light constraints; typical delays are on the order of 1.5-2 nanoseconds per foot. Even signals which stay on a single pc board can easily have inter-chip propagation times of 3-4 nanoseconds; inter-board signals obviously can have much longer transmission times.

2.5 Noise Considerations

The term *noise* is generally used to refer to signal perturbations which displace a voltage at a node from the node's expected value. Offsets in a node's voltage can change the delay associated with changing the logical state of the node; in this way, most noise signals can be considered to introduce some distortion of the delays present as well as the voltage levels present. This delay noise or phase noise is of more interest in this thesis than voltage offsets but voltage noise is usually easier to think about and to quantify accurately. The discussions of noise topics which occur at various points will use occasionally both types of noise under the assumption that there really is no difference between the noises other than in the interpretation of the manifestation of the distortions.

2.5.1 Sources of Noise

Noise can be grouped into two main classes: *Transient* and *Steady State.*

Transient Noise Sources

Fast switching times for output signals require large current surges to charge the large external capacitances; the bonding wires and package leads act as inductors through which this current must flow resulting in significant voltage drops. Switching several outputs simultaneously can result in noticeable noise on the power rails due large power supply current surges. Similarly, the current required to switch internal nodes must often flow through polysilicon or diffusion interconnect causing noticeable voltage drops. Closely spaced parallel wires and wires which cross form coupling capacitances; when one wire switches states, the other wire will tend to move also especially if the second wire is in a high impedance state. Since the largest capacitances couple nodes to the substrate, precharging large arrays can move the substrate voltage due to coupling to the substrate. Alpha particles can induce large charge fluctuations which can temporarily or per-

manently change the states of nodes. Temperature variations can arise due to sudden changes in the type computation being preformed or the amount of communication which is occurring. Temperature induced variations will occur much slower than the other transient noise perturbations. Differences between frequency generators can result in the injection of a continuous phase skew between signals which are controlled by the clocks.

Steady State Noise Sources

Processing variations result in device characteristic shifts which are equivalent to steady state voltage shifts. Steady-state noise voltages also result from operating temperature variations and manufacturing variations in discrete components and pc board characteristics which cause power supply offsets and voltage level variations for signals which depend on resistor dividers.

So far all of these noise sources have been discussed as voltage variations; it is also possible to model noise sources as causing phase variations. Given the definition of phase margins, any perturbation which delays an edge or changes the edge's rise or fall time will cause a shift in the phase margin of the signal with respect to other signals. Modeling noise as phase variations is much more useful for evaluating the issues important in high-speed data communication problems than modeling noise as voltage variations.

2.5.2 Probabilistic Model for Noise

Many of the noise sources just described can be modeled accurately but the complexity of such modeling makes a simple model desirable[3]. A simple probabilistic model is proposed in [14] which will be utilized here.

By studying a logic family and systems employing the family over a period of time[4], the noise in the system can be characterized by a plot

[3]Dynamic RAM designers routinely model all of the types of noise described above but at a great expense in design and simulation time.

[4]Or more realistically by performing a very careful analysis of the nature of the noise sources

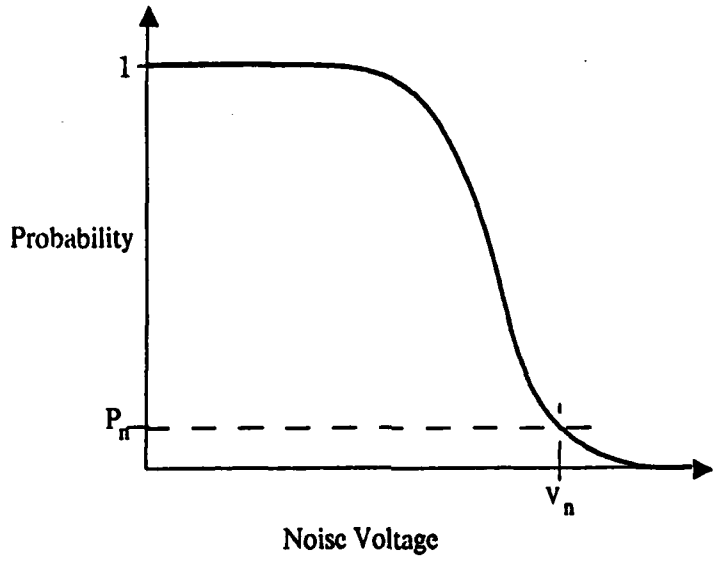


Figure 2.8: Probability vs. Noise Voltage

such as the hypothetical one shown in Figure 2.8. Any given value of noise voltage, v_{n_i}, or phase noise, t_{n_i}, will occur with some probabilities, $P_{n_v}(v_{n_i})$ and $P_{n_t}(t_{n_i})$.

Given such probability functions and the desired probability of noise-induced failure, the designer can determine what worst-case noises any node must be able to tolerate, $v_{n_{MAX}}$ and $t_{n_{MAX}}$. The designer must then insure that all of the logic gates' static and dynamic characteristics provide voltage and phase margins greater than the worst case noises. These worst case noise specifications will be used to simplify all subsequent discussions involving noise considerations.

2.5.3 Static Noise Immunity—Static Noise Margins

The independent specifications for a logic family's input and output volt-

---

once.

ages, provide a convenient mechanism for defining the relative noise immunity of the logic family. The maximum amounts of noise the gates in a family can tolerate on HIGH and LOW signals are given by the noise margins, V_{NMH} and V_{NML} respectively[12]. These margins are:

$$V_{NMH} \equiv V_{OH} - V_{IH} \tag{2.7}$$

and

$$V_{NML} \equiv V_{IL} - V_{OL}. \tag{2.8}$$

As long as the noise margins are greater than the worst case noise, $V_{NMH} > v_{n_F}$ and $V_{NML} > v_{n_F}$, then the probability of the circuit failing will be acceptably low.

The specification of the logic levels and noise margins is fairly arbitrary and even the choice of v_{n_F} is somewhat arbitrary. This does not mean that these will not affect the performance of the logic gates designed to meet the specifications; it can be shown that the design of restoring logic circuits involves a fundamental tradeoff between noise margin and the delay of the circuit [14]. Due to this coupling of the circuit's noise margins and delay, the choices of logic levels would be much more tightly linked to the maximum allowable noise levels than is indicated in this presentation.

2.5.4 Phase Jitter Immunity—Dynamic Phase Margins

The static transfer function given, $v_{out} = H_S(v_{in})$, is a voltage-to-voltage relationship; as such it is possible to check the consistency of the specifications by interpreting the output voltages as input voltages and insuring the that resulting output voltages would be meet the specifications for output voltages. The noise margins are exactly the amount the output voltages could be degraded and still produce legal output levels when applied to a gate. The same is not true of the dynamic transfer function, $v_{out} = H_D(T_{PM}$; the output specification is given a voltage while the input specification is given as phase margin.

Defining a phase margin to phase margin specification would allow defining phase noise margins directly in terms of the transfer function. Because phase margins are only defined at the inputs to clocked gates and because there will almost always be varying amounts of logic between successive clocked latches defining a general phase-to-phase transfer function would be difficult and of minimal utility. Instead, the maximum allowable phase noise, $t_{n_{MAX}}$, will be used as the desired phase noise margin. The system designer must insure that all phase margins exceed the t_S and T_H by $t_{n_{MAX}}$ in order to achieve a suitably low probability of failure.

Chapter 3

Dynamic Delay Adjustment

In order for a digital system to be reliable, the design must either prevent metastable states from occurring or at least make their probability of occurrence so low as to be ignorable. The desire to use a simple synchronous design model but still have a high clock frequency makes it empirative that the two-sided delay constraints, Equation 2.6, be satisfied. In systems built from TTL and ECL components, the amount of circuitry required had a strong affect on the final performance and cost of the system; the design time required was also important but could be amortized over a large number of systems. Therefore in such systems, it made sense to expend the design time required to satisfy all of the delay constraints.

With the advent of VLSI technologies, some of the tradeoffs are changing. Now circuitry is relatively cheap and design time has become a much larger factor in the final success of a machine. The approach presented here reflects this fact. Dynamic Delay Adjustment (DDA) is a circuit based technique for doing exactly what the name implies, dynamically adjusting delays to satisfy the synchronous delay constraints. By embedding knowledge of how to properly satisfy delay constraints in a circuit that can be replicated thousands of times with no additional design expense, all of the delays in a system can be automatically and continuously adjusted.

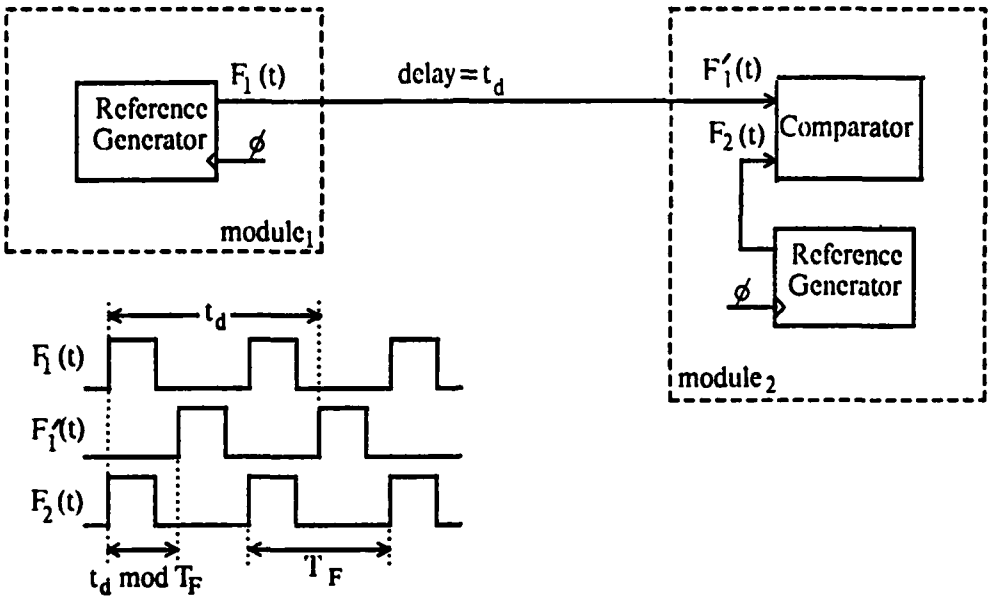


Figure 3.1: Phase Shifting of a Periodic Signal Due to Transmission Delay

3.1 Variable Delay Phase Adjustment

The concepts behind the Dynamic Delay Adjustment synchronization technique are illustrated in Figure 3.1 in which a periodic signal, $F_1(t)$, which has a period of T_F is being generated in one module and transmitted to another module over a path which has some delay $t_d \geq T_F$. If the transmitted signal, $F_1'(t)$, is compared with a signal $F_2(t)$ which is being generated locally and is supposed to be identical to $F_1(t)$, $F_1'(t)$ will appear to be shifted with respect to $F_2(t)$ by $t_d' = (t_d \bmod T_F)$ due to the periodic nature of $F_1(t)$:

$$F_1'(t) = F_1(t - t_d) \equiv F_1(t - (t_d \bmod T_F)).$$

This is hardly a new observation, designers regularly use this fact in the design of clock distribution systems for synchronous machines in order to insure that clock signal transitions occur at exactly the same time in all portions of the machine.

The two sided delay constraints developed earlier also used this fact to allow delays on data lines of greater than the clock period even though the

data signals would not be truly periodic; in this case the designer had to determine the number of clock cycles the data would spend traversing the wire and design the logic to take the delay into account.

Most computers are constructed as many independent modules, in some cases there may be modules that are *activated* by requests from other modules and are simply idle when no messages are being received. In such cases, the exact number of clock cycles messages require to go from one module to another may not be critical[1].

This characteristic led to the idea of using a circuit similar to the one shown in Figure 3.2(a) as a way of automatically adjusting the phase of incoming signals to insure that the signals met the delay constraints of the input circuitry. This synchronizer employs a technique which will be called Dynamic Delay Adjustment (DDA), for the obvious reason that the synchronizer is based on the idea of dynamically adjusting the delay seen by the data signal in order to insure that no synchronization errors occur.

3.2 Operation of the DDA Synchronizer

The heart of the synchronizer is the variable delay line; for a system with a clock period T_C, the delay line must be capable of providing delays which vary from 0 to T_C seconds. The variable delay line is used to adjust the phase of the incoming signal before the signal is sampled by the latch; the rest of the synchronizer must insure that the output of the delay line does not violate the setup and hold requirements of the latch.

The comparator performs a correlation between the delayed signal and the reference signal being generated by the reference generator; this correlation produces a correction signal which is to be applied to the variable delay. The filter adds an integral component to the correlation signal in order to make the system more stable and easier to control. The Change-Enable signal lets the internal chip circuitry control the characteristics of the filter in

[1]This is especially true of many multi-processor architectures.

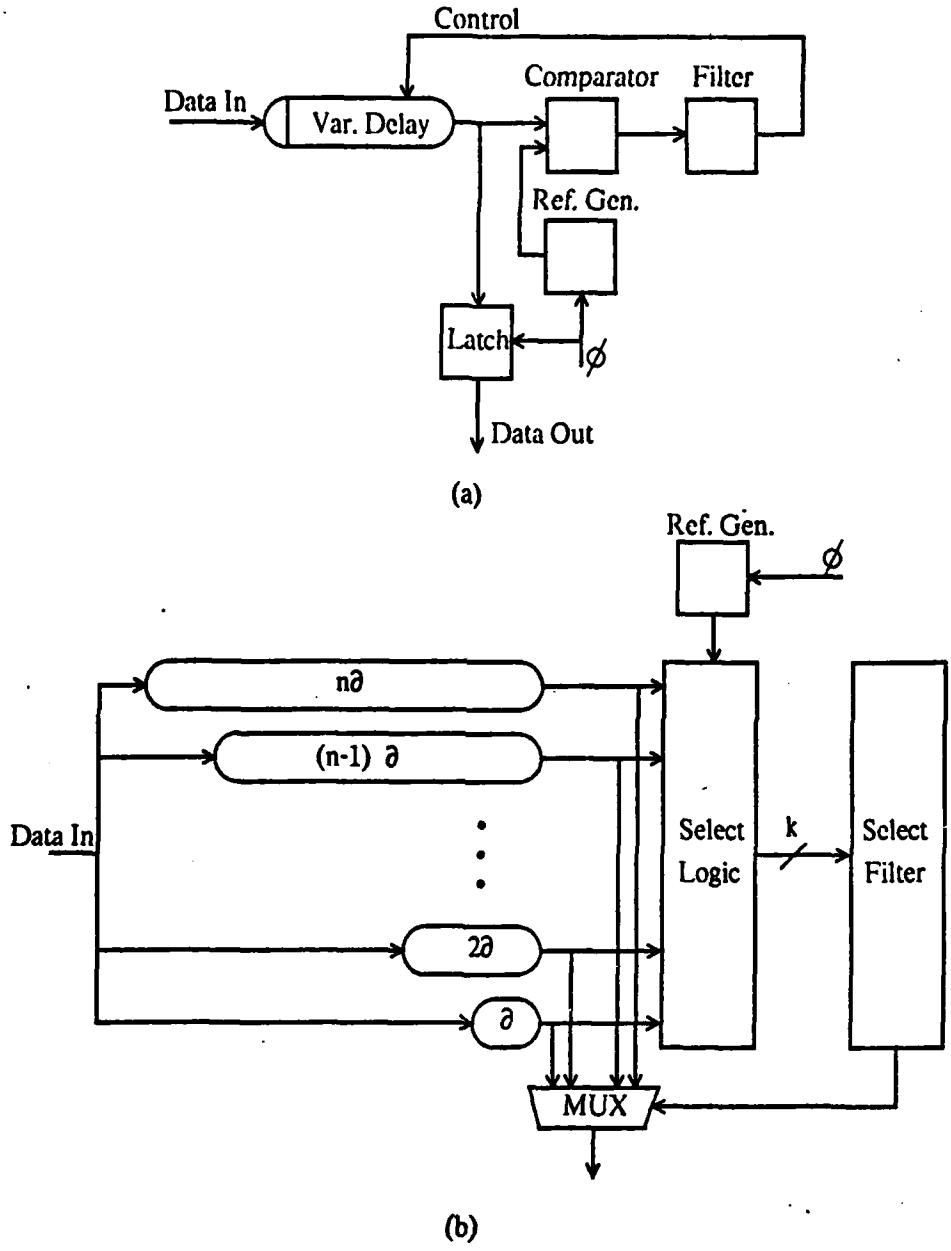


Figure 3.2: Block Diagram of the Dynamic Delay Adjustment Circuit and One of the Common Variations

order to prevent changes in the delay at critical times.

3.2.1 Multiple Delay Variation

Figure 3.2(b) shows a modification of the DDA circuitry just discussed which will be used in both of the implementation strategies discussed. In this variation the variable delay line is emulated by generating several delayed versions of the input and then selecting among them[2]. Provided the delay values are spread evenly over the required T_C range, one of the delay lines should provide a delay that satisfies the delay constraints. This is not a perfect emulation since the multiple delays only provide a limited set of discrete delays. This imperfect emulation is attractive due to the difficulty of generating a very short, easily variable but also predictable and drift free delay in MOS technologies.

An important point to note regarding the multiple-delay version of the DDA synchronizer is that some accuracy has been sacrificed by providing only a limited number of discrete delays in place of one continuously variable delay. Figure 3.3 compares the discrete and continuous adjustment approaches; in either case, the goal is to place the input transitions as far from the latching clock as possible in order to achieve the maximum noise immunity. The ideal phase adjustment would result in the sum of the delay, t_d, and the original phase margin, t_{PM}, being equal to $T_C/2$.

The plot in the Figure assumes there are 4 delays with values spread evenly between 0 and T_C and shows how the phase adjustment can vary from the ideal by as much as $T_C/4$ seconds. In the general case in which there are n delays, the discrete delays may reduce the phase noise margin of the synchronizer by as much as T_C/n. As a result, this technique can only guarantee a minimum phase noise margin of:

$$t_{PM} \geq \frac{n-2}{2n} T_C. \tag{3.1}$$

[2]The delay lines outputs will also be referred to at *taps* for short; this convention alludes to the fact that the various delays could be generated by taking taps off a single analog delay line.

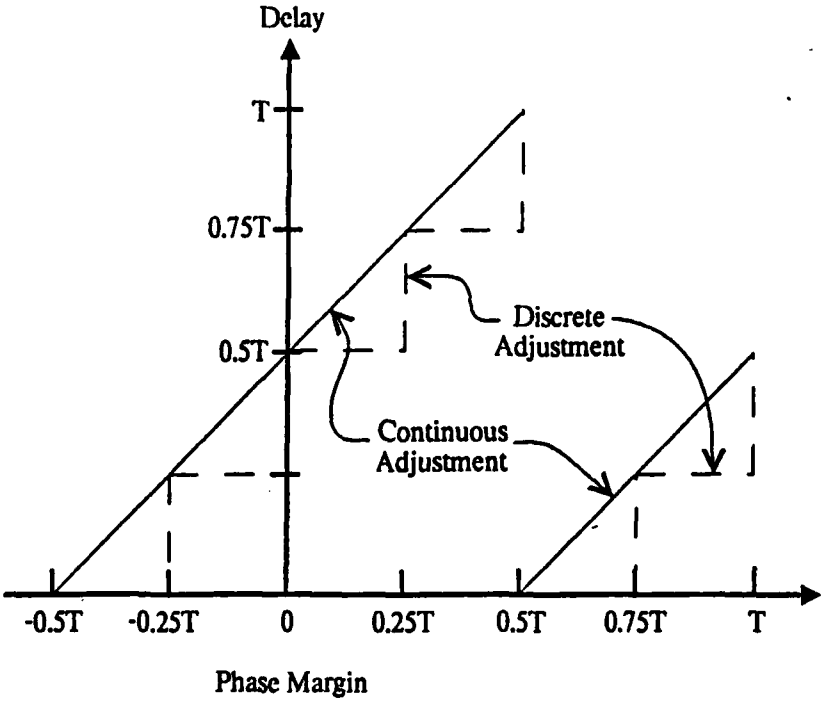


Figure 3.3: Comparison of the Phase Adjustment Provided by the Continuous Delay and the Discrete Delay DDA Synchronizers

This minimum noise margin will also be reduced further if the delay values are not spread evenly over the clock period. The minimum guaranteed phase margin can be found by summing all combinations of $(n-2)/2n$ adjacent delays; the phase margin can not be guaranteed to be larger than the minimum of these sums[3].

3.2.2 A PLL In Disguise?

The description of the DDA synchronizer's operation makes the circuit sound like a Phase Locked Loop (PLL) and basically the DDA synchronizer can be considered to be a PLL. The function of the DDA circuit is different from that of a typical PLL in a very important manner.

In most cases, the function of a PLL is to accept an incoming asynchronous signal and phase-lock onto the clock frequency embedded in the signal in order to synthesize a clock signal with the same frequency and phase shift as a clock signal which had been sent from the transmitter along the same path as the data signal. This synthesized clock signal can then be used to extract the data from the input reliably. This lets the receiver adjust for variations in both frequency and phase but it does not solve the synchronization problem since the phase of the decoded data with respect to the local clock will still be unknown.

The DDA circuit on the other hand assumes that the data frequency is very well controlled and only the phase is unknown. By changing the phase of the incoming signal to be in-phase with the local clock, instead of adjusting a clock to the incoming phase, the DDA synchronizer can address both the decoding problem and the synchronization problem.

3.2.3 Communication Protocol Assumptions

The input is assumed to be an unencoded, serial bit stream with a frequency equal to the local clock frequency; also, it is assumed that on top of

[3] Assuming the delay variation is small enough that all sums of $(n-2)/2n$ adjacent delays are less than $T_C/2$ and all sums of $((n-2)/2n)+1$ adjacent delays are greater than $T_C/2$.

the serial bit stream a message-based communication protocol is being used. With this type of protocol, all inter-chip communications are done by inserting the information to be transmitted into a message which has additional information which enables interpretation of the message by the receiving chip. Any message may be requesting data, returning results, signalling completion of a computation or performing some other function.

In most cases the sending chip will transmit one message and then either wait for a response or go on to perform some other function. There will almost always be some amount of unutilized time between messages if only because the receiving chip will require some amount of time to generate a response[4]. Between messages, an *idle* pattern is transmitted, the idle pattern can be any type of periodic pattern and may be chosen for a variety of reasons, low power consumption or ease of generation and decoding for example. The receiving chip watches the input for a break in the idle pattern which signals the start of the next message. Sections 3.3.2 and 3.4.2 will discuss how the idle pattern can effect the performance of the synchronizer.

3.3 A Digital CMOS DDA Synchronizer

MOS technologies have the nice property that on-chip characteristics track well; device parameters may vary slightly from one portion of the chip to another but they do not vary greatly. The same can not be said for variations from chip to chip; parameters can and do vary noticeably from chip to chip even for wafers fabricated at the same time. These variation makes designing any components which have *absolute* specifications, e.g. voltage references or absolute delays, very difficult.

The approach described in this section skirts this issue somewhat; the problem of generating the delays accurately is hidden by making some assumptions about what types of clock signals are available. Justification of

[4]Bidirectional lines and long serial messages may allow the receiver enough time to formulate the start of a response before the first message is finished and begin transmitting as soon as the line can be reversed. In such cases the between message interval may approach 0.

this assumption will be left to Section 3.4.3 which discusses the issues associated with generating and controlling the clock signals.

A block diagram of the digital DDA synchronizer would be almost identical to the multiple delay version shown in Figure 3.2(b). The only difference at this level would be the absence of a reference signal generator block; the data is assumed to be simple, unencoded data and as such the reference is just the local clock.

The synchronizer described in this section has been implemented as a 3μm CMOS circuit in order to demonstrate the feasibility of this approach. A simple test chip has been fabricated and tested; Section 3.6 discusses testing related issues and results. The synchronizer as presently implemented is not useful other than as a test vehicle for the basic idea; due to lack of time and effort and to some basic misconceptions and oversights during the design phase, the circuitry is much more complex than necessary and requires too much area to be widely used.

3.3.1 Digital Delay Lines Using Overlapping Clocks

Figure 3.4 shows the details of the digital delay lines and the associated clock signals used to generate the delays required by the DDA synchronizer. It should be noted that although 4 clock signals and 4 delay lines are shown, this is not intended to indicate that 4 is the only number of clocks and delays that will work. Any number of delays can be used; 4 delays provided a reasonable tradeoff between noise immunity and circuit complexity in the first implementation.

The circuitry required is fairly simple, consisting of cascaded dynamic register cells similar to the dynamic latches discussed in Section 2.1. The real heart of the delay lines is the set of clock signals used to clock the latches; the clocks are somewhat unusual for MOS circuits because they are not non-overlapping. Ideally, the clocks consist of one phase of the local clock and 3 delayed versions of that clock with the delays arranged such that their falling edges are spaced evenly across the clock period. As will

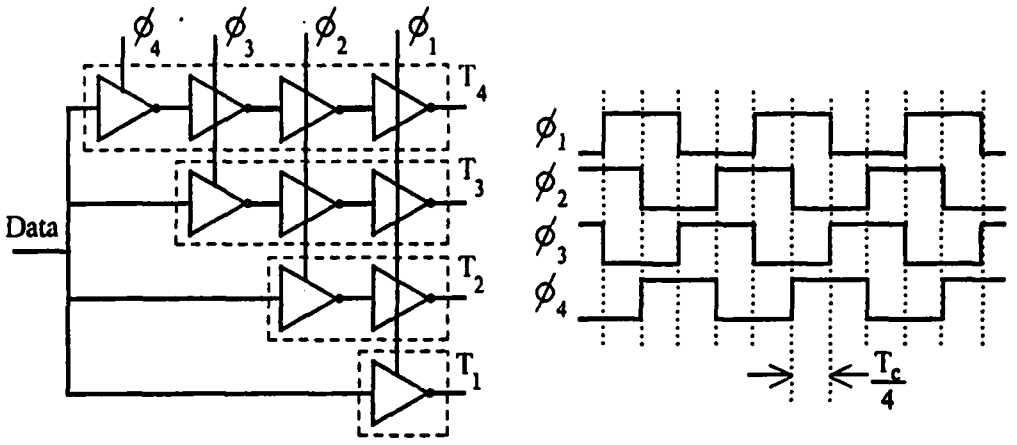


Figure 3.4: The components and clocks used in implementing the digital delay line for the DDA synchronizer

be seen, the important characteristic of each clock is where its falling edge occurs with respect to the local clock; taking the falling edge of ϕ_1 to be the end of the clock cycle, the clock phases are numbered in reverse order from the order in which their falling edges occur within the clock cycle.

Describing the delay lines is complicated somewhat by the 2 different interpretations which are possible for the operation of the dynamic latches. Section 2.1 described the dynamic latches as being level-triggered rather than edge-triggered. This description makes it possible to view the cascaded latches as acting like delay lines which delay input transitions while the first latch's clock is LOW but do not delay transitions otherwise. This interpretation of the delay lines is motivated by the desire to emulate the actual analog delay called for by the DDA synchronizer in Figure 3.2(a).

A slightly more accurate way to describe the latches' outputs is *falling edge latched*. With this description, the first latch in each line can be viewed as sampling the input signal at the falling edge of the latch's clock. The n different delay lines therefore sample the input signal n different time during each clock cycle. The additional latches following each of the sampling latches are needed to delay the samples taken early in the clock cycle until the final sample is taken. The delaying action allows the selection circuitry

to access all of the samples at the same time; also by having the output of each delay line driven by a ϕ_1 latch, the output of the synchronizer can be guaranteed to look almost the same regardless of which of the delay line outputs the MUX is choosing.

Simplification of the Delay Line Circuitry

Interpreting the delay lines as simply oversampling the input signal rather than actually delaying the signal opens the door for some simplification of the delay lines. Figure 3.5 shows 2 ways in which the delay line circuitry can be simplified; all of these examples use 4 delay lines but similar approaches could be used for any number of delays.

The circuit in Figure 3.5(a) uses 2 less latches than the original delay lines. The two internal ϕ_2 latches could be removed due to the non-overlapping nature of the ϕ_1 and ϕ_3 clocks[5]. The ϕ_3 latch following the ϕ_4 latch can not be removed since doing so would allow transitions occurring while both ϕ_4 and ϕ_1 are HIGH to propagate through both latches.

By modifying ϕ_4 to make ϕ_4 non-overlapping with respect to ϕ_1 the extra ϕ_3 latch can be removed also. The resulting delay line circuitry and the modified clock signals are shown in Figure 3.5(b). This simplification of the delay lines would complicate the clock generation somewhat since ϕ_2 and ϕ_4 are no longer normal non-overlapping clocks.

A third simplification can be applied in addition to either of the 2 approaches just described. A slight modification to the interface between the delay line outputs and the selection circuitry can make it possible to replace the 4 ϕ_1 latches by a single latch placed after the MUX. In the present implementation, the inputs to the selection circuitry are latched by a divided down version of ϕ_3 whose falling edge is aligned with ϕ_3's falling edge. This approach would latch the selection circuitry inputs with with a divided down version of ϕ_1; this will provide the same inputs to the selection circuitry

[5]This approach assumes the clock signals are generated using the technique proposed in Section 3.4.3 which results in $n/2$ pairs of non-overlapping clock signals.

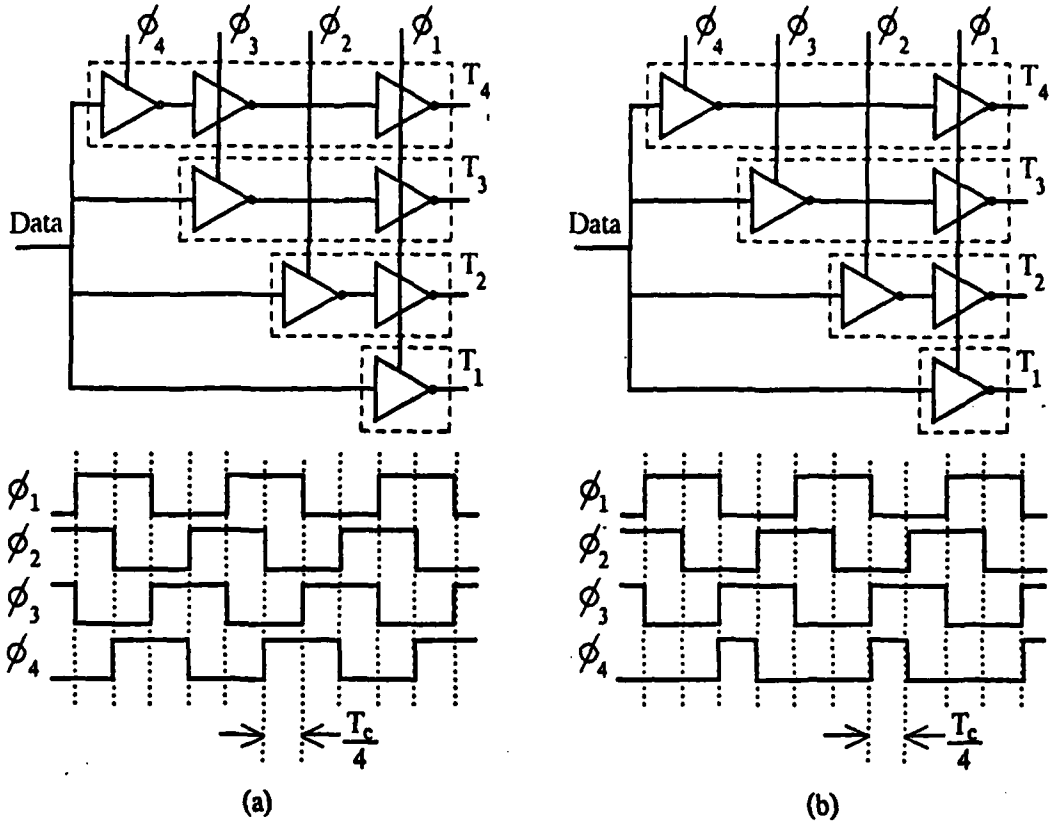


Figure 3.5: Two Strategies for Simplifying the Circuitry Required by the Digital Delay Lines.

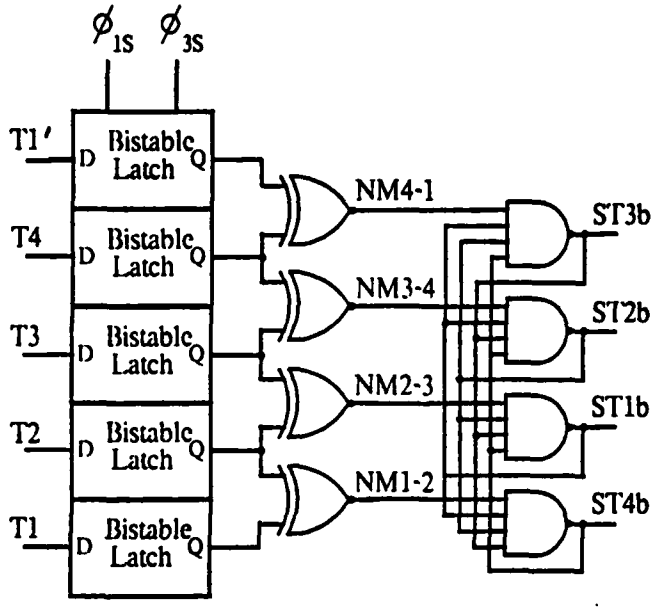


Figure 3.6: Circuitry for Performing the Delay Selection by Transition Position Detection.

only the inputs are accepted half a cycle earlier. The dynamic latch after the MUX latches the output to insure the output obeys the local clocking protocol.

3.3.2 Choosing the Proper Delay by Transition Position Detection

The discussion in this section pertains equally well to any synchronization scheme which generates several delayed versions of the input and then selects between the different versions. The technique presented here is remarkably simple and is based on detecting when transition occur by comparing the outputs of adjacent delay lines.

Figure 3.6 shows the circuitry used to decide where within the clock cycle the data transitions are occurring. The decision circuitry consists of 5 static, bistable latches, 4 exclusive-or (XOR) gates and 4 4-input NAND gates. The latches, and all of the control circuitry, are clocked by ϕ_{1S} and ϕ_{3S} which

are divided down versions of the non-overlapping local clocks, ϕ_1 and ϕ_3; these slower clocks will also be referred to as the control clocks. Figure 3.6 also shows the relationship between the local and control clocks; the figure does not give an exact frequency ratio, $r_C = f_{\phi_{1S}}/f_{\phi_1}$, since this ratio will be determined by the characteristics of the bistable latch as described in Appendix A.

Operation of the Selection Circuitry

The static latches are used to insure that the outputs of the delay lines are at logically legal states before the delay selection circuitry tries to interpret the outputs. The last section claimed that only 4 delay lines were used, so why are there 5 static latches? As will be seen shortly, although only 4 delays will be considered for use in adjusting the phase of the input, the 4 delays are not sufficient for making a delay selection if the input stream is not known to have a data transition in every sample cycle. The fifth delay line samples the input with ϕ_1 and then delays the sampled value for 1 complete clock period; this extends the range of sampling times to a full clock cycle compared to the $3T_C/4$ range available with only 4 delay lines.

At the falling edge of ϕ_{3S}, the inputs to the bistable latches should be the values sampled during the previous local clock cycle; further, these values will have had $T_C/2$ seconds to propagate through the bistable latches. During ϕ_{1S}, the bistable latches will settle to legal states; if there was a data transition during the sampled data cycle, it must have occurred in between the falling edges of two of the clocks[6]. A data transition will divide the taps into two contiguous groups; one group whose members latched a HIGH value and another group which latched a LOW value. By XORing each pair of adjacent taps, the transition point can be easily detected.

At most one of the XOR gates will have a HIGH output. If there are no HIGH outputs, then there could not have been a data transition; this is

[6]This discussion is ignoring the possibility of metastable outputs by assuming the bistable latches have been given sufficient time to settle to a valid logic value with a probability extremely close to 1.

the situation the fifth delay line was added to catch. Had there only been 4 samples, the selection circuitry would be unable to distinguish between a cycle in which there was not a transition and one in which the transition occurred between ϕ_4 and ϕ_1. If one of the XOR gates has a HIGH output, the transition occurred between the falling edges of the two clocks which latched the input to generate the values input to the XOR gate. This places the transitions within the interval between two clocks, which is as accurate a placement as this scheme can produce.

Assume for example the data transitions are occurring between ϕ_1 and ϕ_2 with the result that taps 2, 3, 4 and 5 are always equal while tap 1 was different. The XOR gate between tap 1 and 2 would be the only gate with a HIGH output indicating that the transition was between ϕ_1 and ϕ_2 as required. The proper choice of delay is that delay that which will place the latching clock signal as far from the data transitions as possible. Due to the limited accuracy of this technique, the data transition could actually be arbitrarily close to either ϕ_1 or ϕ_2; so either ϕ_3 or ϕ_4 would be an equally valid choice as the sampling clock.

The outputs of the XOR gates are sufficient to indicate where the transitions are occurring and also which delay line should be chosen. It still remains to explain the function of the 4 NAND gates which are included in the selection circuitry. Basically the NAND gates perform almost no useful function and should have been left out of the implementation. The gates are interconnected so that 3 inputs to each NAND gate are the outputs of the other 3 NAND gates and the fourth input is one of the XOR gate outputs. The resulting circuit is called a 4-flop and it has the characteristic that as long as at least one of the inputs is HIGH, then only one of the 4-flop's outputs will be LOW.

This has the somewhat useful function of insuring that if for some reason more than one XOR gate produces a HIGH output only one of the outputs of the selection circuitry would be HIGH. There are 2 ways more than one XOR gate could produce a HIGH output, either 2 transitions occurred on the input during the sample period due to distortion of the input signal or

there was a noise glitch somewhere in the circuitry. If the data signals are distorted enough that 2 of the XOR gates are going HIGH due to there being 2 transitions during the sampling period then the synchronizer will probably not function properly anyway; if the extra HIGH signals were produced by a noise glitch then the selection filter should remove the glitch not the selection circuitry.

How this 4-flop came to be in the design is clear in hindsight but how the fact that it performed no real function was not noticed is harder to explain. Basically, the 4-flop grew out of an earlier plan to use an analog correlation technique which would have been used in conjunction with the analog implementation of the synchronizer which is briefly described in Section 3.5. The analog correlation technique required a 4-output mutual inhibition gate which is what the 4-flop is. When the correlation technique was changed the 4-flop was not removed.

3.3.3 Some Additional Problems with the Implementation

Continuing in the spirit of the discussion of the 4-flop, some other misconceptions concerning the operation of the synchronizer which were in place during the implementation should be discussed before the rest of the implementation is presented. Figure 3.7 shows an expanded view of the circuitry used to implement one bistable latch and one XOR gate. The latch is pretty much as expected except that the Q-bar output is followed by 2 inverters which drive the inputs to the XOR gate. This approach of following the dynamic latches by buffers which drove the inputs to gates rather than directly using the outputs of the latches was used in several places in the synchronizer. This approach was taken due to a misconception concerning how the control clocks would be shaped; the original waveform assumptions are shown in Figure 3.7 as ϕ_{1S} and ϕ_{3S}. Because the dynamic latches must interface with the faster data handling section, it was assumed that ϕ_{3S} could only be HIGH for the last $T_C/2$ seconds of ϕ_{3S}'s period. Further it was as-

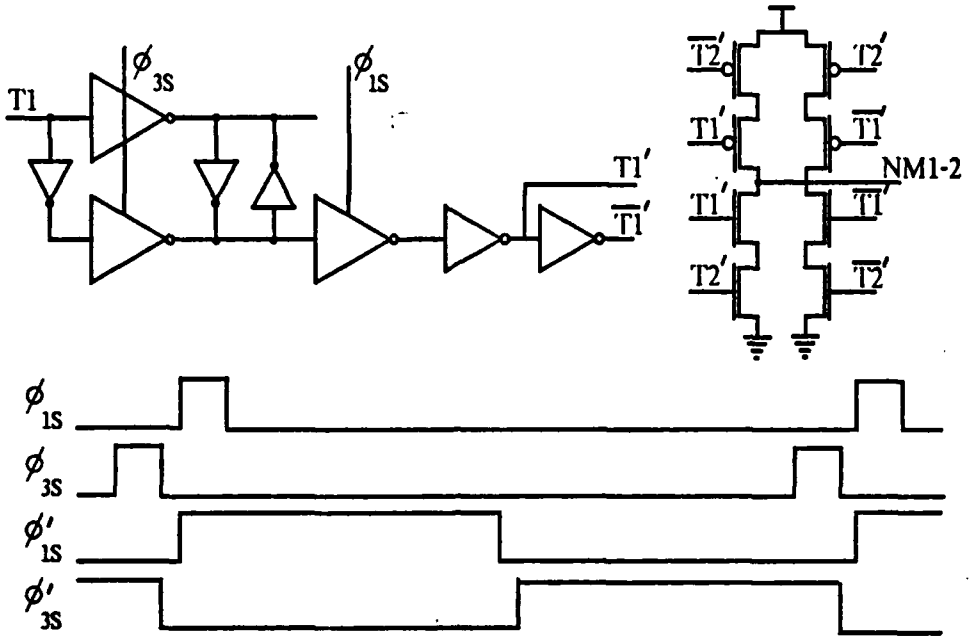


Figure 3.7: An Illustration of the Circuitry Overhead Introduced by the Clocking Misconception.

sumed that ϕ_{1S} would be generated in the same manner as ϕ_{3S} and would therefore only be HIGH for $T_C/2$ seconds also[7]. Such small duty cycles severally restricted the fanout of any of the dynamic latches; rather than worry about the clock period being limited by a dynamic latch with too much load the outputs of the latches were buffered.

The actual clocks used in the synchronizer are similar to the ϕ'_{1S} and ϕ'_{3S} signals shown in the Figure; the need for the bistable latches to interface with the data handling section only requires that the falling edge of ϕ'_{3S} occur before the rising edge of ϕ_1. There is no limit on the duty cycle. The longer duty cycle would alleviate the fanout problem associated with the dynamic latches making it possible to remove the extra inverters.

Another related problem with the current design grew out of the failure to predict the performance of the bistable latches before designing the

[7]It should already be obvious that not enough time was spent thinking out the design of the synchronizer before the implementation was committed to silicon.

implementation circuitry. Since the settling time which the latches would require was not calculated ahead of time, the length of the control clock's period was not known; in order to not have the clock period set by the rest of the control circuitry[8], the design was pipelined. Breaking the control section into 3 stages complicated the design procedure, adding conceptual and circuit complexity without adding any real gain in performance.

Most of the room for improvement in the synchronizer implementation arises from these 2 wrong assumptions. Section 3.4.5 will estimate how much the area could be reduced by *unpipelining* the design and slowing down the control circuitry.

3.3.4 The Selection Filter

The data signal will always possess a certain amount of phase jitter due to noise in the system. Such noise is typically assumed to have a normal distribution and a zero mean value; given this type of distribution there is some probability that large noise *spikes* will occur occasionally even if the rms value of the noise is very small. Although such noise signals may occasionally cause a bit-time to be so short or long that sampling errors occur, the selection circuitry must be able to insure that the effects of the noise do not continue to effect the performance of the synchronizer after the noise has subsided.

Consider for example a situation in which the data transitions are occurring in between ϕ_3 and ϕ_4 but are actually very close to ϕ_3. The synchronizer will choose tap 2 as the delay in this case. Let a large noise signal be injected for a short time which moves a single bit's transition from the nominal location near ϕ_3 to after ϕ_4. The XOR gates will interpret this as meaning the proper tap is now tap 3; were the tap selection switched to tap 3 due to this single sample, problems could arise when the noise subsided and the transitions returned to their nominal location near ϕ_3. Metastable states could be sent into the internal chip logic causing errors to occur until the

[8]This would not necessarily be bad anyway.

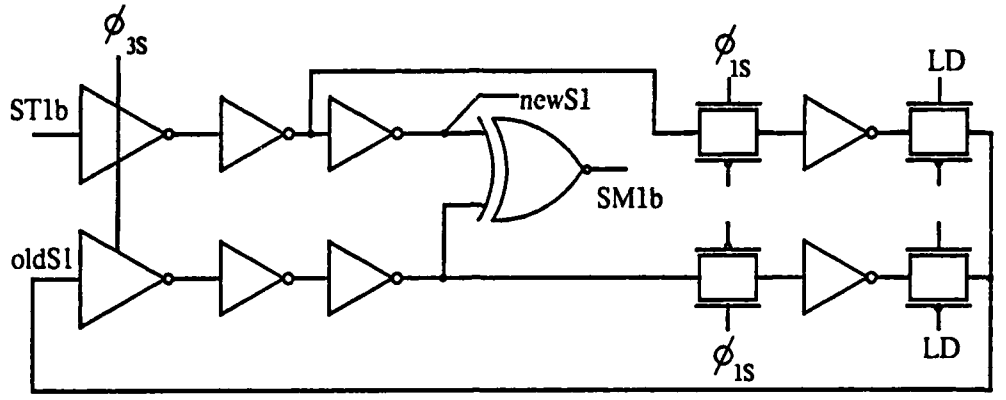


Figure 3.8: The Challenger/Candidate Selection Comparison Circuitry.

selection was changed back to tap 2.

For this reason, the selections generated by the XOR gates are *low-pass filtered* to remove transient delay choices due to transient data phase changes. The technique used to do this filtering was to require the XOR gates to make the same selection several more times than any other selection was made before any change in selection was made.

At any point in time there are 3, possibly different, selections active in the synchronizer. The oldest and most important selection is called the *incumbent*; this is the selection being used to control the MUX and is held by the selection latch. The next oldest selection is called the *candidate* and the newest selection is the *challenger*, these last two selections are the internal state of the filter and the input to the filter (output of the 4-flop) respectively.

The circuitry used to hold one bit of the candidate selection and perform one bit of the Candidate/Challenger comparison is shown in Figure 3.8. The left hand portion of the circuitry is basically 2 dynamic latches clocked by ϕ_{3S} whose outputs are buffered and drive an XOR gate to produce the Selection-Match-1 (SM1) signal. The outputs of the two dynamic latches

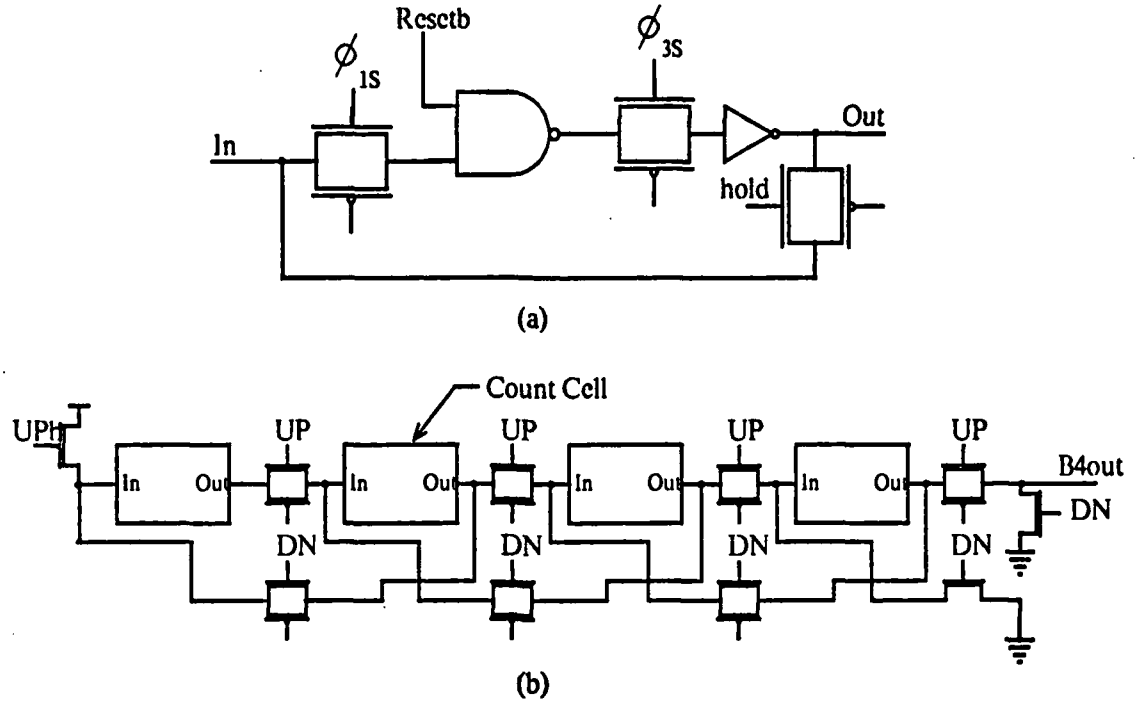


Figure 3.9: The UP/DOWN/HOLD State Register for the Selection Filter.

are also fed into a selector which is controlled by the Load (LD) signal. If LD goes HIGH, the Challenger selection is loaded into the lower latch and becomes the new Candidate; otherwise LD will be LOW and the Challenger will be fed back in to restore the latch for the next comparison.

Also involved in the filter is a 4 bit UP/DOWN/HOLD shift register which acts to keep score for the selection filtering process. The circuitry for one bit of the register is shown in Figure 3.9(a) and the manner in which 4 bits are cascaded to form the entire register is shown in Figure 3.9(b). The state of this register is determined by the number of bits which are currently in the HIGH state. On every cycle one of the 3 control signals, UP, DN or HOLD must be raised and the register's state will change in the appropriate manner, increasing in value, decreasing in value or holding the same value.

Filter Operation

On each cycle of the sample clock, a new challenger selection will be generated by the selection logic. This selection will be compared with the candidate selection, if the two match then an UP signal is sent to the shift register increasing the registers value. If the selections do not match, a DN signal will be generated to decrement the register's value. If the data period being examined did not contain a data transition, then none of the Select inputs will be valid and the HOLD signal will generated.

If the register's value is 0 then the challenger selection is loaded in as the new Candidate. If the register's value is 4 then the Candidate is considered to be a valid selection; the signal indicating the Candidate has been qualified is gated by a control signal from the internal chip logic known as Change-Enable (CE). The CE signal indicates that the internal logic is not extracting information from the data stream and a change of delay selection is acceptable. Assuming CE is HIGH then the Candidate becomes the Incumbent.

Figure 3.10 shows the circuitry used to generate the control signals needed by the selection comparison circuitry and by the UP/DOWN/HOLD register. The 4 new-Selection signals (newS1 etc.) are NORed together to form the HOLD signal while the 4 SM signals are NANDed together with the externally provided Resetb signal to form the an intermediate control signal SMallb. SMallb (Selection-Match-all-bar) and its complement, SMall, are then NORed with Hold to form the UP and DN signals respectively. Finally, the SMallb signal is NANDed with the negative true version of the least significant bit of the register, B0b, to generate the LDb signal.

The filter is actually broken into 2 pipeline stages and all of these operations do not occur simultaneously. Explaining the pipelined operation would not add much useful information and the next implementation will probably not be pipelined, so a combinatorial explanation of the filter was given.

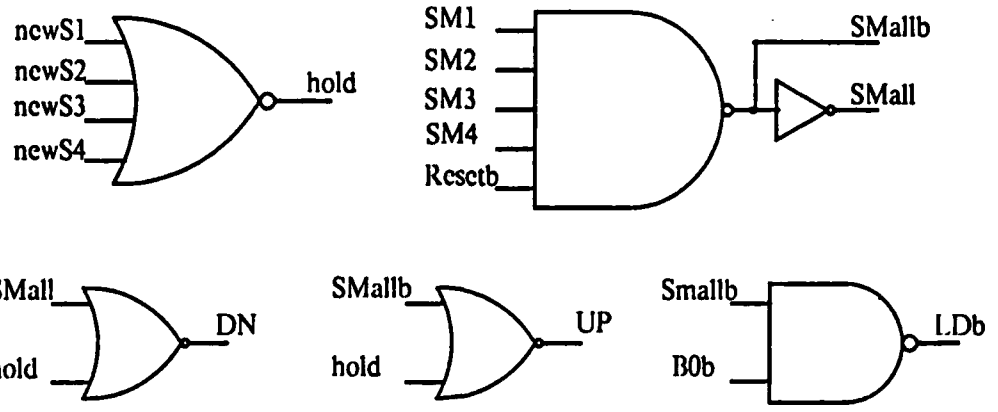


Figure 3.10: Circuitry for Generating the Selection Filter Control Signals.

3.3.5 The Selection Latch and the Delay MUX

The circuitry used to implement one bit of the selection latch and delay MUX is shown in Figure 3.11. The fourth bit of the UP/DOWN/HOLD register is NANDed with the Change_Enable signal to generate the Selectb signal. This signal controls a dynamic latch whose data input is one bit of the Candidate selection from the selection filter. As long as Selectb is LOW the Incumbent selection is updated to be equal to the Candidate selection; when Selectb goes HIGH, the Incumbent is latched and will no change.

The Incumbent selection in turn controls a dynamic latch whose data input is the output of the delay lines; the outputs of the 4 latches which are controlled by the 4 Incumbent bits all directly drive the output of the synchronizer.

3.4 Additional Design Considerations

3.4.1 What Limits the Data Rate?

Since the aim of the DDA synchronizer was to allow high speed data transfers between MOS chips, the bottom line in evaluating the synchronizer

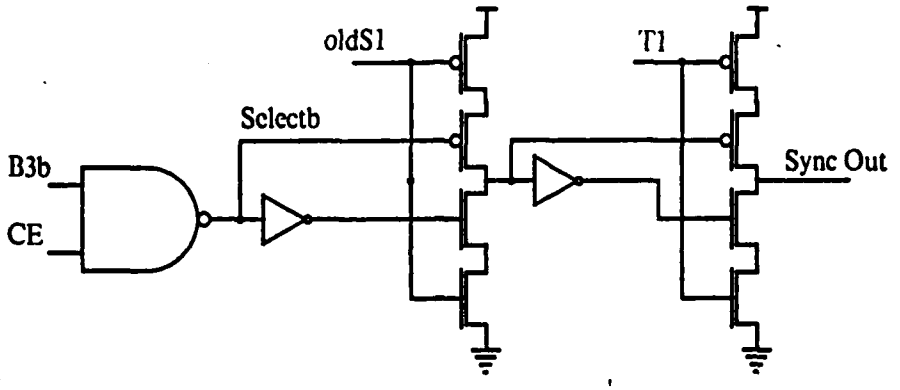


Figure 3.11: The Circuitry for One Bit of the Selection Latch and the Delay MUX.

implementation is the maximum reliable data rate. In discussions regarding speed, the synchronizer can be divided into two distinct sections, the low speed control section and the high speed data handling section.

The data rate is dependent solely on the clock frequency of the high speed data handling section of the synchronizer. The ultimate limit on the clock frequency is the speed at which the dynamic latches can turn on, change state and then turn off but the practical limit will turn out to depend on the characteristics of the multi-phase clocks and the noise immunity requirements. The clock rate in the low speed section will depend on the settling time required by the bistable latches but this clock rate has no direct effect on the actual data rate.

If the phase of the input signal were known exactly and there were no noise, then the data rate would only be limited by the setup and hold requirements of the dynamic latch. Assuming the system uses a 2-phase clock and each phase has a duty cycle of $< 50\%$ and assuming that $t_S \geq t_H$, the clock period would have only be limited to $T_C \geq 2t_S$.

As discussed in Section 3.2.1 the phase is only known to within the resolution of the discrete delays, or in this case the resolution of the clock signals, and there is also noise that must be accounted for in order to assure reliable operation. Assuming the minimum guaranteed phase margin of the synchro-

nizer is some fraction of the clock period, $t_{PM_{MIN}} = T_C/k$, and the maximum phase noise is $t_{n_{MAX}}$, then the clock period must be chosen so that:

$$t_{PM_{MIN}} \geq t_S + t_{n_{MAX}}$$

$$T_C \geq k(t_S + t_{n_{MAX}}). \tag{3.2}$$

This relation shows how increasing the accuracy of the synchronizer reduces the overhead which must be introduced into the clock cycle and thereby increases the maximum data rate.

3.4.2 Effect of the DDA Synchronizer on Data Transmission Efficiency

As is the case with most performance measures, the maximum data rate, $f_{D_{MAX}} = 1/T_C$, is only half of the story. The real measure of the synchronizer's performance is the effective data rate, $f_{D_{EFF}}$. Ideally, $f_{D_{MAX}} = f_{D_{EFF}}$; this section will consider how the noise present in real systems limits the amount of useful data which can be transmitted causing $f_{D_{MAX}} > f_{D_{EFF}}$.

The loss of transmission efficiency is caused by the break down of the assumption of an absolutely controlled data frequency. The discussion of noise sources in Section2.5.1 briefly mentioned two types of low frequency noise which can introduce distortions which are, or appear to be, frequency shifts: data dependent thermal variations and frequency shifts due to crystal variations.

Both of these noise sources will be considered in more detail shortly but for now, assume the local clock has a period of exactly T_C seconds but the incoming data signal has a slightly shorter period of $T_C - \delta_C$. The incoming data will appear to be distorted by a continuous noise signal which is injecting a phase skew of $-\delta_C$ seconds per local clock period. This phase noise is cumulative in nature; assuming the input transitions are initially occurring between ϕ_2 and ϕ_3, the phase margin of the input with respect to ϕ_3[9] will decrease by δ_C every clock cycle. The synchronizer will not see

[9]The first local clock which follows the input transitions.

any skew in the input until the input skews completely past ϕ_3. Once the skew reaches these proportions, the synchronizer must be able to respond and change the delay selection before the transitions skew far enough to approach the sampling clock signal.

How quickly the synchronizer can respond to phase skew depends on the characteristics of the selection filter. For instance, the current implementation requires a minimum of 4 cycles of the control clocks to approve a selection following a reset of the synchronizer but would require a minimum of 8 cycles in order to respond to a change in phase. Let N_F be defined to be the minimum number of cycles of the control clock required to change the delay selection; for the current synchronizer $N_F = 8$.

Combining the skew rate, δ_C in seconds per data cycle, and N_F with the ratio of the control clock period to the data clock period, r_C, and the minimum guaranteed phase margin, $t_{PM_{MIN}}$, yields expressions for the maximum allowable skew for a given N_F or the minimum allowable filter delay for a given skew:

$$\delta_{C_{MAX}} = \frac{1}{r_C}\frac{t_{PM_{MIN}}}{N_F} \tag{3.3}$$

$$N_{F_{MIN}} = \frac{1}{r_C}\frac{t_{PM_{MIN}}}{\delta_C} \tag{3.4}$$

For the present implementation $T_{PM_{MIN}} = T_C/4$ and $N_F = 8$; the control clock ratio r_C is not absolutely set but Appendix A suggests $r_C = 12$ to obtain a sufficiently low probability of failure. Using these values, a $\delta_{C_{MAX}}$ can be calculated:

$$\delta_{C_{MAX}} = \frac{1}{12}\frac{T_C/4}{8} \approx 2.5 \times 10^{-3} T_C.$$

Sources of Low Frequency Noise

A brief discussion of how the type of skew just described can arise is in order. There at least 2 types of noise which can cause short or long term frequency skew: frequency variation between the local and remote clocks and thermally induced phase variations.

If a single crystal oscillator is used to generate the basic clock frequency for the entire machine then the data frequency would be exactly the same everywhere. However, in order to increase the reliability of the machine several oscillators might be used. Although crystal oscillators are very accurate and well controlled, no 2 crystals will be exactly the same so some slight frequency variation will occur.

Crystal oscillators with frequency tolerances of 0.01% provide frequency control tight enough to limit the skew rate to $\delta_C = 10^{-4}T_C$ seconds per cycle. This amount of variation is well within the theoretical capability of the synchronizer.

Thermal variations can also produce phase shifts by changing the delay of the off-chip drivers and receivers. Most delays in MOS chips are directly proportional to the electron and hole mobilities of the devices; the temperature dependence of the mobilities can be expressed as:

$$\mu(T_2) = \mu(T_1)\left(\frac{T_1}{T_2}\right)^M \tag{3.5}$$

where T_1 and T_2 are absolute temperatures and M is between 1 and 2, typically taken to be 1.5. For instance, if the chip increased from a room temperature of 20°C to relatively low operating temperature of 50°C, the mobilities would decrease by a factor of 0.86 and the delays would increase by roughly 15%.

Steady state thermal variation can be accommodated by allowing some reasonable warmup time after the machine is initially powered up. There is another type of thermal variation which can not be handled so easily: data dependent thermal variations. The discussion of delay in Section 2.4.2 mentioned that the capacitive nature of most loads in MOS systems introduced an important CV^2f component to the power dissipation. One point that was not mentioned at that time was that the f in the power equation is really the frequency of signal transitions rather than clock frequency. Since the data being transmitted between chips is not encoded in any way, consecutive 1's or 0's in the data will not produce any data transitions so the frequency of transitions will be less than the clock frequency whenever actual

data, as opposed to an idle pattern, is being transmitted. During operation, the transition frequency seen on any individual line will be dependent on the actual data being transmitted and will vary greatly; the power being dissipated by the buffer driving the line will also vary greatly.

The substrate will conduct the heat away from the pad driver but because the power density will be much greater around the pad drivers, the temperature will be higher also. The data dependent fluctuations in the power being dissipated may also cause larger temperature fluctuations around the off-chip buffers than in the rest of the chip. The large delays of the drivers and receivers relative to the clock period make even 10-20% increase in delay an important reduction in the noise margin. No estimate of the δ_C's which can be generated by data dependent thermal variations has been attempted.

When Can Delay Changes Be Made?

Assume that a situation exists in which data transitions are occurring between ϕ_2 and ϕ_3 and the signal is skewing towards ϕ_2. The sampling clock will originally be ϕ_1 but when the input skews past ϕ_2, the selection circuitry will eventually decide to change the sampling clock to ϕ_4. When the switch in sampling clocks occurs, 2 scenarios are possible. Assume first that the MUX switches within one data clock cycle and define t_i to be the time when the falling edge of ϕ_1 occurred which produced the last output bit sampled with ϕ_1. On the next cycle, the output bit will have been sampled by the falling edge of ϕ_4 which occurred at $t_i + T_C/4$ seconds; this sample was actually the same input bit that the last ϕ_1 sample saw. The switch in sampling clocks introduced an extra bit into the data stream; just the opposite will occur for a switch from ϕ_4 to ϕ_1, one bit will be missed.

Another possibility is that the switch will not occur within one data clock cycle but rather within one control clock cycle. In this case, r_C data bits may be missed and/or be output as illegal logic state due to a slow switching time.

For both of these reasons, the internal logic must have some control over

when changes in the delay selection occurs. This was the reason for including the Change_Enable (CE) signal. When useful data is being transmitted the internal logic must lower CE in order to insure that delay changes do not cause extra or missed bits.

The inevitability of losing bits or producing extra ones means that there must be some times when no data is being transmitted so delay selection changes can be made. Further, the idle time must occur frequently enough to insure that the delay selection can always be changed before the signal skew causes phase margin problems; this limits the length of contiguous messages.

The situation is further complicated by the fact that the data within the messages can not be relied on to generate transitions for the selection circuitry[10]. The expressions for $\delta_{C_{MAX}}$ and $N_{F_{MIN}}$ assumed that data transitions were observed on every cycle of the control clocks. A more conservative approach is to assume that no data transitions will be observed by the delay selection logic and then to determine the percentage of the bandwidth that must be allocated to between message idle time in order to insure proper synchronizer operation for some given δ_C, N_F, r_C and $t_{PM_{MIN}}$.

Let k_S be the number of data cycles required for the skewing signal to reduce the phase margin to 0; k_S will be given by:

$$k_S = \frac{t_{PM_{MIN}}}{\delta_C}.$$

Within every k_S data cycles the control circuitry must observe at least N_F data transitions; insuring this requires $N_F r_C$ idle pattern data cycles. These idle pattern bits reduce the effective data frequency:

$$\begin{aligned} f_{D_{EFF}} &\leq f_{D_{MAX}} \left(1 - \frac{N_F r_C}{K_S}\right) \\ &\leq f_C \left(1 - \frac{N_F r_C \delta_C}{t_{PM_{MIN}}}\right). \end{aligned} \tag{3.6}$$

Assuming the thermally induced skews will be less than the frequency difference induced skews, the current synchronizer has a maximum effective

[10]Most data transmissions will contain some type of parity or checksum information to help guard against transmission errors; these techniques will insure that the data signal has some minimum transition frequency, f_t. This minimum frequency will translate into some minimum average frequency of *observed* transitions.

data rate of:

$$\begin{aligned} f_{D_{EFF}} &= \frac{1}{T_C}\left(1 - \frac{12\times 8\times 10^{-4} T_C}{T_C/4}\right) \\ &= 0.96 f_C. \end{aligned}$$

3.4.3 Clock Generation and Control Issues

In the digital implementation presented in this chapter, the delays and the resulting phase margins depend on the relationships between the falling edges of the sampling clocks. Specifically, the time differences between the falling edge of each clock and the falling edge of ϕ_1 will be the delay values, and since only 4 clock phases were used in the actual implementation, $t_{PM_{MIN}}$ would be the smallest time difference between the 4 falling clock edges. The clocks are intended to be spaced exactly $T_C/4$ seconds apart; the actual phase margin will depend on how accurately the clock phases are generated. This section will consider one method of generating the clock edges and the sensitivity of the resulting phase margin to various types of clock distortions.

Figure 3.12 shows one simple method of generating the 4 sampling clocks starting from two clock signals, ϕ_A and ϕ_B [11]. ϕ_A is a single phase clock with a period of T_C and a duty cycle of t_{D0}/T_C; ϕ_B is exactly identical to ϕ_A but is offset (delayed) with respect to the ϕ_A by t_{OFF} seconds. The input clocks are inverted to generate $\overline{\phi}_A$ and $\overline{\phi}_B$; the delay of the inverters is assumed to be t_{INV}. Each pair of cross-coupled NOR gates splits one of the input clocks into two non-overlapping clock phases; the delay of the NOR gates is assumed to be t_{NOR}. The resulting 4 clock phases are also shown in Figure 3.12.

Taking the rising edge of ϕ_A to be $t_0 = 0$ seconds, the falling edges of the 4 clocks will occur at:

$$\begin{aligned} t_1 &= t_{OFF} + t_{D0} + t_{INV} + t_{NOR} \\ t_2 &= t_{D0} + t_{INV} + t_{NOR} \\ t_3 &= t_{OFF} + t_{NOR} \\ t_4 &= t_{NOR} \end{aligned}$$

[11] This section does not discuss problems associated with the generating the complements of the clock.

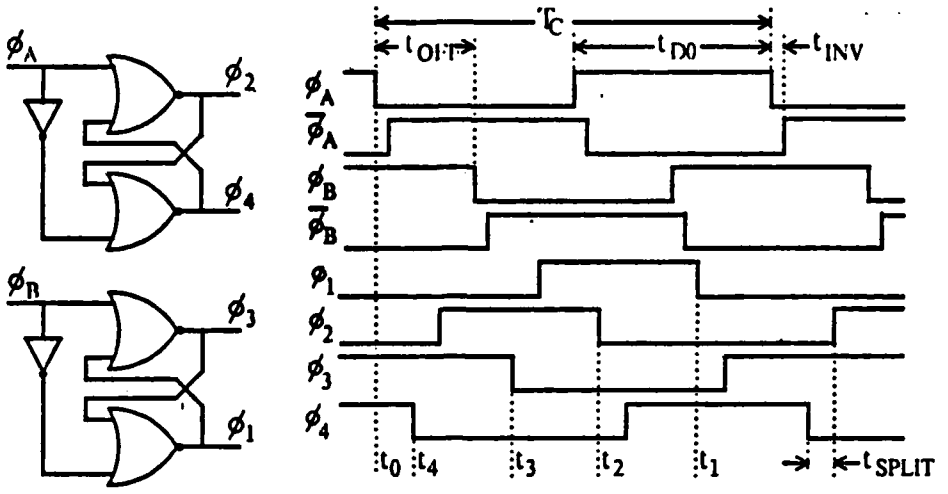


Figure 3.12: One Simple Method for Generating the 4 Sampling Clocks from 2 Input Clocks.

The ideal values for t_{D0} and t_{OFF} are $T_C/2$ and $T_C/4$ respectively; the actual values will vary from the ideals somewhat resulting in $t_{D0} = T_C/2 + \delta_{D0}$ and $t_{OFF} = T_C/4 + \delta_{OFF}$. By plugging these values into the equations for t_1-t_4 and then taking the differences between adjacent signals[12]:

$$t_{PM_{MIN}} = \min \left\{ \begin{array}{l} \frac{T_C}{4} + \delta_{OFF} \\ \frac{T_C}{4} + \delta_{D0} - \delta_{OFF} + t_{INV} \\ \frac{T_C}{4} + \delta_{OFF} \\ \frac{T_C}{4} - \delta_{D0} - delta_{OFF} - t_{INV} \end{array} \right\}. \qquad (3.7)$$

This shows that the initial clocks' duty cycle and the offset between the clocks both directly effect the synchronizer's minimum phase margin.

Generating ϕ_A and ϕ_B

The generation of ϕ_A and ϕ_B must be based on some technique which does not require accurately controlled MOS gate delays. One approach to the problem is to take advantage of the availability of predictable delays in

[12]The difference between ϕ_1 and ϕ_4 is found from $t_4 - t_1 + T_C$.

bipolar technologies; very accurately controlled ECL and TTL delay lines are available. A single crystal oscillator could be used to generate ϕ_A which would be distributed to all of the system; then an ECL delay line could be used to delay ϕ_A to generate ϕ_B *locally*[13]. The two major problems with this approach are controlling the distribution of ϕ_A and ϕ_B on each board and more importantly controlling the distortion of the signal during the ECL-to-CMOS level conversion stage. This approach would also limit the flexibility in choosing the clock frequency during debug and testing stages of the design.

A second possibility for generating ϕ_A and ϕ_B is also based on delaying ϕ_A by a well controlled amount; rather than use an active delay line the passive delay inherent in the pc board traces could be used. The transmission speed along the pc traces depends only on the electromagnetic properties of the pc board materials and traces. ϕ_A could be distributed to every chip and the ϕ_A and ϕ_B inputs could be interconnected in such a way that ϕ_A will take $T_C/4$ seconds to travel from the ϕ_A input to the ϕ_B input. No additional circuitry would be required to generate ϕ_B. This approach is as feasible as the first; the main drawback of this approach is that once again the signals will be subject to distortion during the signal conversion and buffering stages. Also, there would be no way to vary the frequency once the pc boards were fabricated.

A third technique would be to distribute a clock signal with a period of $T_C/2$ and divide the signal down to the proper clock frequency on-chip. The main advantages of this technique are that only a single I/O pin would be needed for the clock signal and no extra off-chip circuitry or pc board complexity would be required. The additional on-chip circuitry would be minimal since some type of frequency division is going to be required anyway to generate the slower clocks for the control section of the synchronizer. Generating all of the clocks by frequency division followed by phase splitting might make the alignment of the slow and fast clock edges would be easier

[13]Locally would probably mean one delay line per board or at least one per several chips.

also. This technique would allow varying the clock period to facilitate debug and testing without limiting the final speed of the system.

The frequency division could be performed in such a way that the rising edge of the clock caused ϕ_A transitions and falling edges caused ϕ_B transitions. This would give an offset between ϕ_A and ϕ_B equal to the duty cycle of the original clock signal which ideally would equal $T_C/4$ but this offset would still be susceptible to distortion of the original clock by the level conversion circuitry.

One argument against the on-chip division technique is that if the clock division circuitry can operate at double the data rate, the data period could be cut in half. While the clock frequency is ultimately limited by the switching speed of the dynamic latches, the more realistic limit will be the delay through 2 or more stages of arbitrary logic. The frequency division parts of the circuitry would be fairly simple so it is feasible that operation at double the clock rate is possible but no design has been attempted to confirm this claim.

None of the clock generation techniques are immune to distortion of the sampling clock signals due to distortion of the duty cycles of, and offset between, ϕ_A and ϕ_B. The frequency division approach is more flexible than the other approaches in terms of changing the frequency, but the off chip ECL delay line approach would be the simplest in terms of design.

3.4.4 Non-Random Phase Jitter

The discussions of noise have assumed the phase noise distortions all consisted of basically 3 classes of noise: fast transient jitter, low frequency phase skew and dc phase offsets. The synchronizer and data transmission protocol were designed with these three types of noise in mind. There is another type of phase noise which the current implementation does not try to handle — phase jitter caused by distortion of the duty cycle of the data signal.

A CMOS inverter with equal width pullup and pulldown will distort a

data signal due to the differences between the characteristics of the n- and p-type devices. Such distortions can be designed around to a certain extent by adjusting the ratio of the device widths in the gates and by other more involved techniques. Such adjustments only work perfectly for one particular set of device characteristics and the characteristics of the n- and p-type devices do not necessarily track each other over process variations. Any pair of chips may be fabricated at different times and the resulting variation in the sending chips output buffers and the receiving chips receiving circuitry can result in different phase margins for rising and falling transitions; the setup and hold requirements for the latches are also slightly different for falling and rising transitions.

Future implementation may have to take this distortion into account in order to avoid having the synchronizer fail to make a delay choice due to seeing different phases for rising and falling edges.

3.4.5 Area Requirements for DDA Synchronization

Although the first goal of the DDA synchronizer was to allow high speed data transmissions directly between MOS chips, the second goal was almost as important: keep the technique and circuitry simple enough to allow utilizing the synchronizer on **every** wire in a system. This would mean placing as many as 50-100 synchronizer on a single chip; the area required must still be only a small percentage of the entire chip area in order for such wide scale utilization to be possible. Determining a maximum size beyond which the synchronizer would no longer be useful is fairly arbitrary; the criteria used here is that the synchronizer must be no large than the high speed pad driver being used.

Given this restriction, the current implementation is almost 7 times too large to be used widely. The 3μm synchronize implementation is 1900 by 900 microns compared to an experimental 50 MHz low power pad driver which is 375 by 675 microns. This section will discuss how the next implementation can attack the problem of shrinking the synchronizer by a factor of 8.

The area utilized by the synchronizer is divided between 5 major components: delay lines, selection logic, filter logic, MUX circuitry and empty space. A rough accounting of the space used by the different components gives the following break down:

Delay Lines:	1.0×10^5 sq. microns
Selection Logic:	2.2×10^5 sq. microns
Filter Logic:	6.5×10^5 sq. microns
MUX Circuitry:	1.0×10^5 sq. microns
Empty Space:	6.4×10^5 sq. microns

An immediate observation is that simply compacting the existing layout to remove as much of the empty space as possible would probably reduce the area by 20-30%. The second observation is that most of the area is allocated to the control circuitry as opposed to the circuitry that actually does the useful work: the delay lines and the MUX. Each of the areas will be considered individually to estimate how much improvement is possible over the existing design; the area reductions listed in the discussion of each section refer to the reduction of the area required for that particular section — not for the entire synchronizer.

The Delay Lines

The description of the delay lines in Section 3.3.1 gave 3 approaches to simplifying the circuitry required for the delay lines. The most promising approach reduced the circuitry from 10 latches to just 4 resulting in an area savings of 60%. One of the 4 remaining latches could also be removed at the expense of a slight increase clock circuitry complexity.

The Control Section

The area required for the selection and filter logic can be reduced at the expense of lengthening the period of the control clock. A major conceptual change would be to *unpipeline* all of the control logic by removing most of the dynamic latches and making the control logic combinatorial. Also, the

transistor sizes used in the control logic could be reduced; the current implementation makes very liberal use of donut transistors in order to increase the speed of the control logic[14].

The effect of the unpipelining will be discussed separately for the selection and filtering logic but the reduction in area due to using smaller transistors should be fairly uniform. Given the current $3\mu m$ design rules, a simple inverter using donut transistors requires roughly 2.5 times as much area as an inverter with only slightly larger than minimum size devices. This is too optimistic of an estimate to apply to all of the control logic; a reduction of the area by a third is more realistic and is the reduction factor that will be used.

The Selection Logic

The main reduction which is unique to the selection logic is the removal of the 4-flop; this alone would amount to a reduction of 30%. Without using a different type of bistable latch, the area required for the latch would be hard to reduce drastically. Replacing the single dynamic latch followed by 2 buffers with 2 smaller buffers directly driven by the static latch would reduce the area by another 15%.

The Filter Logic

The unpipelining process would simplify the circuitry needed for the Candidate/Challenger comparison process; this circuitry presently accounts for roughly 40% of the filter logic. A single dynamic latch to hold the Candidate selection, inverters to generate the complementary signals and the XOR gate should be sufficient to perform the comparison. Given such a drastic cut in the circuitry, only a rough estimate can be made but an area

[14]Donut transistors use a square *donut* of polysilicon set in a large area of active area. The hole in the donut is usually used as the drain of the transistor while all of the diffusion surrounding the donut forms the source. In this manner, a transistor which is more than 4 times the minimum can be formed with no more parasitics than a minimum size device would have.

reduction of 50% seems reasonable provided the device sizes are also scaled down.

A design change that could reduce the filter area significantly would be to encode the selection signals into 2 bits rather than using all 4 bits in the selection filtering process. This would add a little overhead to encode and decode the selection but the circuitry needed for the comparison part of the filter would be cut in half.

The best way to shrink the counter section of the control circuitry is not fully understood at this time. One factor that is still unknown is just how much filtering action is really needed; an area reduction of 50% would be automatic if the N_F delay of the filter were cut in half by shortening the counter to 2 bits from 4. Other options include using a simple binary counter which is reset when the selection comparison fails or some other type of counting technique. An entirely different scheme could greatly simplify the circuitry required. At this time, the only guaranteed reduction would be by reducing the device sizes which has already been estimated to produce a reduction of 33%.

The Delay MUX and Selection Latch

The delay MUX is the only part of the synchronizer besides the delay lines that must be fast; as such reducing the area of the MUX by a noticeable amount will require a different design approach. As such no estimate of the area savings will be made.

The Select Latch does not have to run fast provided the chip circuitry knows not to use the output of the synchronizer for r_C clock cycles after raising Change_Enable. The circuitry of the latch could be simplified somewhat to reduce the number of additional buffers required; combined with reducing the sizes of the transistors, the net area savings should be around 50% for the latch. Since the latch currently accounts for half of the MUX/Latch circuitry, the total reduction will be around 25%.

Design Changes

The incremental changes suggested for each section of the synchronizer, and the general shrinking of the control circuitry will go a long ways to getting the reduction in size that is needed; however, achieving a reduction by a factor of 7 or more will difficult without major changes in the structure of the synchronizer. The discussion of the maximum effective data rate pointed out that even with skewing of the data due to unmatched clock frequencies there each synchronizer only needs to examine the input transitions for a small portion of the time in order to monitor the phase and make corrections to the delay selection. This opens the possibility of sharing a single block of control circuitry between 2 or more synchronizers; since the control circuitry is the largest portion of the synchronizer, amortizing the circuitry over several input ports would produce a big savings in area.

Conclusion

Totalling all of the proposed simplifications and reductions without sharing the control sections produces an estimated reduction in the area of the synchronizer to around 6.3×10^5 square microns; this is a reduction of less than a factor of 3. Sharing the control logic between 2 synchronizers would boost the reduction factor to 4.5 times and then cutting the filter in half would produce a cumulative gain of area reduction by a factor of 5.5.

Further sharing of the control logic and encoding the selection bits coupled with a more clearly thought out design in general should eventually reduce the area to a manageable size.

3.5 An Analog Approach

An alternative to the approach just described was also investigated; this alternative was more analog in nature and was based on the approach used to construct a single-chip Ethernet transceiver[1]. This approach uses a Voltage Controlled Delay Line (VCD) and takes advantage of the good on-chip

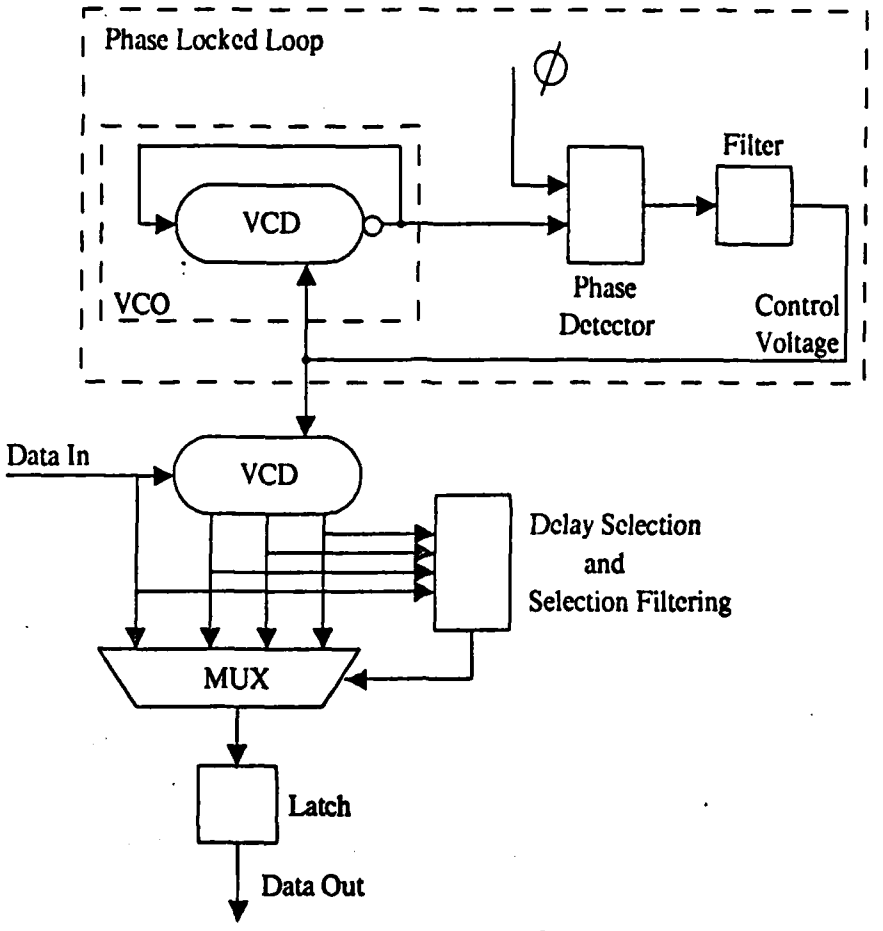


Figure 3.13: Block Diagram of the Analog Based DDA Synchronizer

parameter tracking which MOS technologies provide. The VCD is used to construct a ring oscillator which can in turn be used as the Voltage Controlled Oscillator (VCO) in a Phase Locked Loop (PLL)

A block diagram of a DDA synchronizer using this approach is shown in Figure 3.13. This version of the synchronizer uses the multiple delays variation of the DDA block diagram with the slight variation that all of the versions are generated by a single delay line. The most important circuit components in this version are those which collectively control the delay line, the VCD, the Phase Frequency Detector (PFD) and the Loop Filter (LF).

This approach involves the design of a Phase Locked Loop with all of

the associated complications due to stability issues, frequency acquisition, lock detection etc. Since this was not the approach chosen, a discussion of the issues involved which really did justice to them is not within the scope of this thesis; only a brief description of the operation of the analog DDA synchronizer will be given. The subject of PLL's is treated very thoroughly by Gardner in [10] and [11].

When the computer is first powered up or if the synchronizer is reset, the reference frequency (the local clock probably) will be used to bring the PLL into lock. Assuming the system is designed properly, the feedback control provided by the Phase Frequency Detector (PFD) will gradually adjust V_C until the frequency of the VCO is exactly equal to reference frequency; the phase of the VCO will also probably be adjusted to equal that of the reference signal. Once the PLL is in-lock and depending on whether the ring oscillator's output is divided down or not, the delay of the VCD will be either exactly equal to the clock period or the delay will be some fraction of the period.

The same control voltage is also applied to another VCD which is designed to have identical characteristics to the VCD used in the VCO. In this manner, when the PLL is locked onto the reference frequency, the delay of the second VCD will be known almost exactly[15]. By applying the data input to this VCD and taking taps off the delay line at appropriate points, a variety of delayed versions of the input are generated. The selection of the proper delay could then be done in much the same manner as for the digital synchronizer.

The main reason for not choosing this approach was the difficult of designing a high frequency PLL which would be stable over all the processing variations. High frequency operation would require a very short VCD consisting of a few voltage controlled delay elements; variations in the delay of each element can result in variations by a factor of 2 or more in the basic frequency of the VCO. Accommodating this wide variation requires a delay

[15]The designers at Xerox claimed to be able to control the delay to within a 0.1% error.

element which is fairly sensitive to the control voltage but a high-gain VCO makes the entire system less stable. This situation is further compounded when the synchronizer will be fabricated through MOSIS since the process variations can only be guessed at. The digital synchronizer seemed much more likely to be functional even given wide processing variations.

3.6 Testing Results

The CMOS implementation of the DDA synchronizer has been fabricated through the MOSIS fabrication service[6]. The testing of the chips is the subject of this Section.

The testing process was intended to be a 2 step process: low speed testing to insure the chips were functionally correct followed by high speed testing to determine how fast the chips were. Instead a four stage process was required: functional testing of the first chips, internal probing to determine why the first chips did not work, functional testing of the second set of chips and finally high speed testing of the second set of chips[16].

3.6.1 Generating the Clocks

The most difficult part of both the low speed and high speed testing was generating the clock signals. In both cases, the four clock phases were generated by splitting 2 offset clock phases as is described in Section 3.4.3. For the low speed clocks, High-Speed CMOS NOR gates and NAND gates were used to generated ϕ_1—ϕ_4 and their complements; ϕ_1 and ϕ_3 were used directly as the control clocks without doing any frequency division since the clock frequency was already so low. For the high speed clocks, ECL NOR gates were used to generate ϕ_1—ϕ_4; the control signals were generated by dividing down ϕ_B with a 4-bit ECL binary counter and then using NOR gates to split the clock into ϕ_{1S} and ϕ_{3S} phases. The ECL level clocks were converted to full CMOS levels by using ECL-to-TTL translators with

[16]This last step is still in progress.

the +5 Volt power supply set to +6 Volts in order to achieve almost full CMOS level swings. The translators were designed for differential inputs so the complementary versions of the clocks were generated by driving the negative true input of the translators instead of the positive true input.

A Tektronix 9100 DAS logic analyzer was used to provide the ϕ_A and ϕ_B signals for the low speed circuitry. The DAS can provide output signals up to a clock frequency of 25 MHz; by making one cycle of ϕ_A equal to 8 DAS cycles generating the offset between the clocks was simply a matter of offsetting the ϕ_B output from the ϕ_A output by 2 cycles. The high frequency versions of the ϕ_A and ϕ_B clocks were generated using a Tektronix PG501 250 MHz pulse generator as a trigger source for another PG501 and a PG507 50MHz generator.

The most questionable aspect of the technique used to generate the high speed clocks was the ECL-to-CMOS level conversion process. There are no actual ECL-to-CMOS level converters explicitly available so the choice was to build discrete level converters or try and use the ECL-to-TTL translators that are available; the latter approach was chosen. Because the TTL outputs will only pull up to within 1—1.5 volts of the HIGH voltage rail, the +5 input to the ECL-to-TTL chips was set at approximately 6 Volts. While the low output level of the translators is not 0 Volts, it was low enough that no effort was made to adjust it. The cross-coupled NOR gates are only guaranteed to produce non-overlapping signals directly at the outputs of the NOR gates; the amount of non-overlap time is roughly one gate delay which for the ECL 10KH family of gates is around 1.0—2.0 nanoseconds. The specifications on the propagation delays of the ECL-to-TTL translators is from 1.0—3.6 nanoseconds. Since this is of the same magnitude as the expected non-overlap time some care must be taken to insure that the clocks are still non-overlapping after the level conversion; this is not too difficult since there are 4 translators per package and there are never more than 4 signals whose edges must be carefully controlled[17].

[17]For example, although ϕ_1 and ϕ_3 and ϕ_{1S} and ϕ_{3S} must be non-overlapping, the relation between ϕ_1 and ϕ_2 is not critical except in relation to the noise margin.

3.6.2 Input and Output Signal Generation and Sampling

Besides the clock signals, very few other signals required by the synchronizer: one data input, a reset signal and a Change_Enable signal. The synchronizer produces only 2 outputs: the output data signal and the Select signal.

Once again the slow speed case was straightforward since the DAS could be used to generate all of the input signals and sample the output signals. The generation of ϕ_A and ϕ_B only required a 4 DAS cycle long period for the synchronizer signals, but by slowing the signals down to 8 DAS cycles, the phase of the input could be easily controlled; transitions could be placed in the center of any of the 4 phase *windows*.

This slower signal also made it possible to tell which delay line was being chosen by the selection circuitry without having access to any of the internal signals. Figure 3.14 illustrates the series of 4 short pulses that was used to determine which delay is being chosen; the important characteristic of these signals is that each pulse is overlaps one of the falling edges of the sampling clocks. In the Figure, the initial data pattern was transitioning between the falling edges of ϕ_3 and ϕ_4 but it is impossible to tell which delay is being chosen simply by looking at the output because the output is latched by ϕ_3. Because the short pulses will only be *seen* by the clocks which go LOW during the pulse, only one of the 4 pulses will be seen by the synchronizer for any particular choice of delay. The fact that ϕ_2 was being chosen for the sampling clock is reflected by the output going HIGH after the ϕ_2 pulse but not after any of the other 3 pulses. In this manner the operation of the synchronizer could be fully checked at low speeds without needing any internal signals.

The DAS was not fast enough to provide the clock signals for the high speed tests, but it was fast enough to generate the data and control signals. One of higher order bits of the ECL counters used to generate the offset clocks was used as an external clock for the DAS; this provided an easily

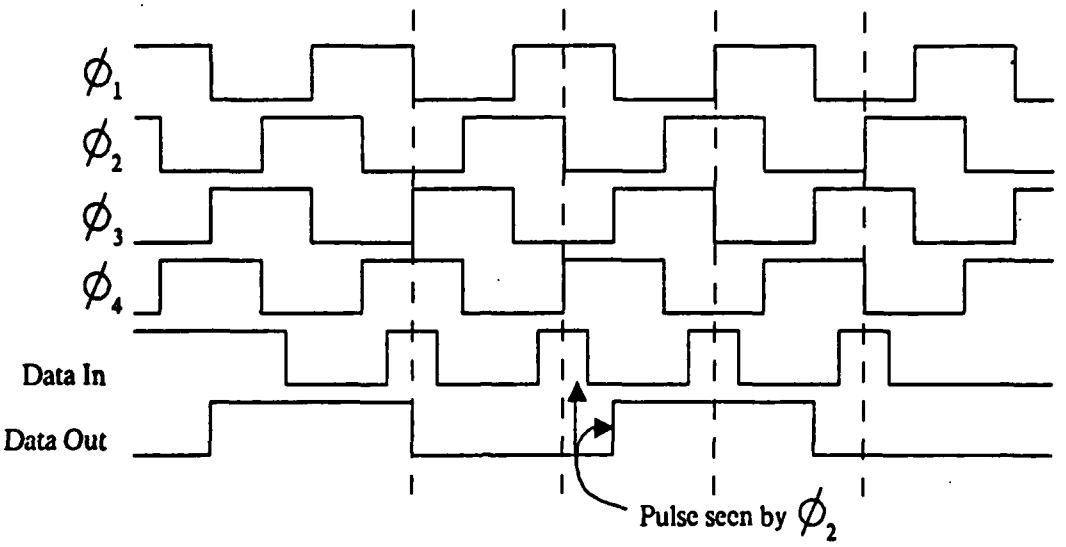


Figure 3.14: The Data and Clock Signals Generated Used to Verify the Operation of the Delay Selection Circuitry.

controlled source of signals which were at least roughly aligned to the synchronizer clocks. The phase of the data signal could not be varied by the DAS with this technique so a voltage controlled delay line was used[18]; by varying the voltage applied to the delay line, the phase of the input could be varied widely.

3.6.3 Results: Chip Set #1

A test chip containing 2 synchronizers was submitted to MOSIS for fabrication in September 1984. The first set of wafers failed the acceptance tests at MOSIS so the run had to refabricated, chips from the second run were received in late January 1985. Once a simple test structure was put together, only a few days were required to determine that of the 8 chips received (16 synchronizers) only 7 synchronizers showed any signs of doing anything. Of those 7 only 4 seemed to be close to working properly and even these 4 seemed to only be able to choose one particular delay line — the line whose input was sampled by ϕ_4.

[18]Such a delay line just *happened* to have been fabricated on the test chip.

The next 3 weeks to a month were spent trying to determine why the chips did not work; this process required using microprobes to examine internal nodes within the synchronizer. The layout of the synchronizer had included overglass cuts at selected places inside the synchronizer logic; these cuts were to serve as probe openings to allow looking at the chosen internal nodes by using a very low capacitance probe. Due to an oversight at the time of the layout, the internal overglass cuts were not made large enough and as a result the openings did not get cut through to the underlying metal lines during processing. Without pre-cut openings, an ultrasonic cutter had to be used to remove the overglass; this process was much more difficult than it appeared and a number of initial attempts resulted only in ruined chips and probe tips.

When internal nodes were finally examined, an interesting phenomenon was observed at the output of one of the buffers driving the transition detection XOR gates: the signal was only remaining HIGH for half the clock period. This pointed to a dynamic charge storage problem at the outputs of the dynamic latches; in order to confirm this idea, the leakage current of the junctions needed to be observed at least indirectly. The first technique used to see if excess leakage was occurring was to examine the dc standby current of the chip - it turned out to be on the order of milliamps for some of the chips. Then to insure the diagnosis was right, the junction characteristics were examined directly using the probe station. The input protection resistor and diodes were used as sources of isolated junctions whose characteristics could be measured; the results of the measurements are shown in Figure 3.15. The figure plots the junction current vs. junction voltage where a positive voltage should forward bias junction and turn on the junction diode while a negative voltage should reverse bias the diode. The plot shows that the reverse biased p+ diffusion-to-substrate junctions were extremely leaky. This was accepted as enough explanation of why the first chips did not work.

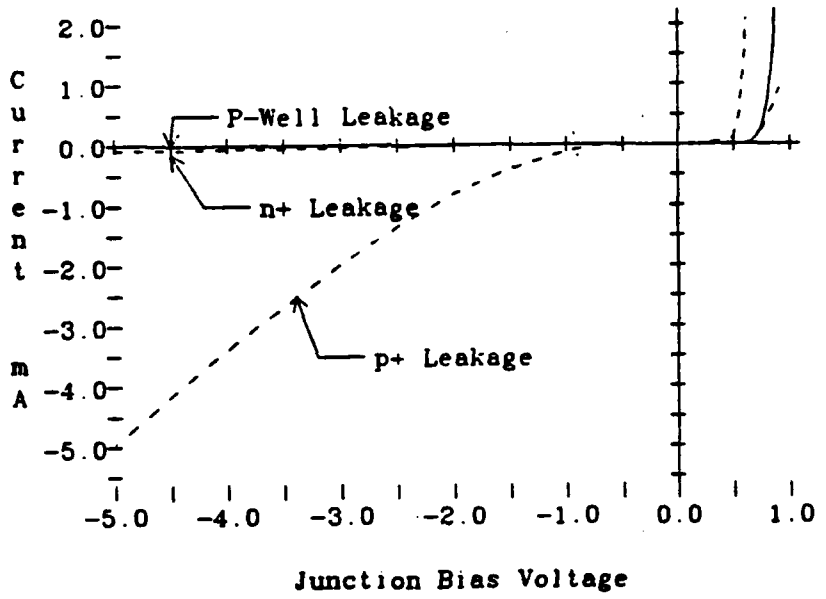


Figure 3.15: The Current-Voltage Characteristics of the PN Junctions of the First Test Chips.

3.6.4 Results: Chip Set #2

A second test chip had also been submitted for fabrication; the chip was actually a test chip for some new pads but a synchronizer was added as an after thought. The functional testing of the second chips went much smoother; within 2 days of receiving the chips their full functionality had been verified. The synchronizer made the proper delay selection for all phases of the input signal and when the phase was changed the synchronizer responded by changing the delay selection properly.

The high speed testing of these functional chips is still proceeding.

3.7 Conclusions

This thesis has presented a circuit based synchronization technique designed to allow high speed data transmission directly between MOS chips in a synchronous system without a detailed analysis of the actual delays involved. The technique provides phase jitter immunity of close to 1/4 of

a clock period; greater immunity is available at the expense of more clock generation circuitry. An implementation of the technique presented has been fabricated; test results have verified the functionality of the synchronizer and tests to determine the speed limits of the synchronizer are continuing. An estimate of the area requirement indicates that the chip area could be reduced enough to make the overhead associated with the synchronizer acceptable.

Appendix A

Linear Analysis of the DDA Flip-Flop

A number of studies have examined bistable latches implemented in technologies ranging from bipolar MSI and LSI[4] to NMOS LSI and VLSI implementations. However, up to this point no studies have been found in the literature which considered CMOS bistable latches. This section will analyze the particular type of latch used in the CMOS implementation of the DDA synchronizer.

Figure A.1 shows the gate-level diagrams of two types of bistable latches; (a) is probably the most commonly implemented form in bipolar and NMOS technologies while (b) is the type chosen for use in the DDA synchronizer. The latches operate in similar manner and have similar characteristics with regard to failures; the discussion in this Appendix will be oriented towards the DDA latch with comments concerning the other type of latch where pertinent. There are a number of different ways to implement bistable latches; the choice made for this implementation was not completely arbitrary but a more thorough analysis of the alternatives could produce a latch with more desirable characteristics.

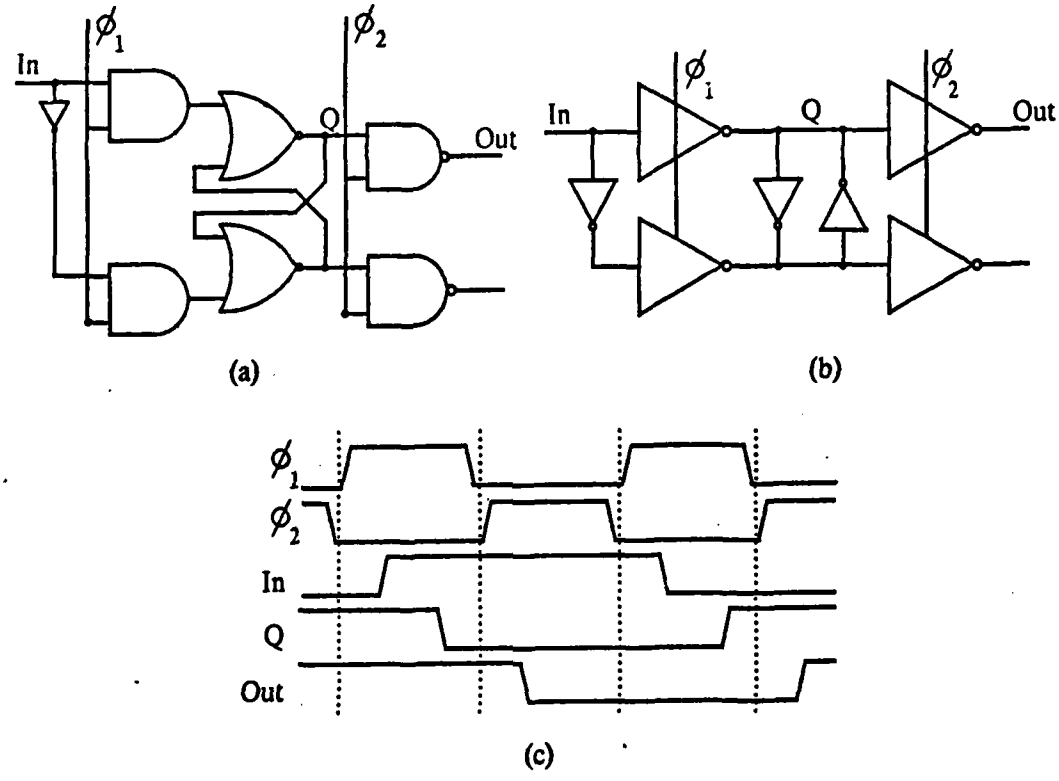


Figure A.1: Two Types of Bistable Latches and the Associated Clocking Signals

A.1 Bistable Latch Operation

Figure A.1(c) shows the timing diagram of the DDA latch under normal operating conditions. When ϕ_1 is HIGH, the transition on the Data input causes a corresponding transition on the Data-bar signal. Together, Data and Data-bar tend to force the outputs of the tristate latches to switch; in order to switch, the tristate latches must overcome the cross-coupled inverters which are trying to hold the latch in the previous state. It is this fighting between the tristate latches and the cross-coupled inverters, that differentiates the DDA latch from the other latch in which the AND gates will naturally switch the latch regardless of relative strengths of the devices[1]. Once ϕ_1 has gone LOW, the cross-coupled inverters are isolated and will settle to one of the latch's stable states. The latch's outputs must be at valid levels before ϕ_2 goes LOW in order to prevent illegal logic levels from propagating through the logic which use the latch's outputs.

Figure A.2 plots the static transfer characteristics of the latch's cross-coupled inverters. The plot show how the feedback present in the latch results in a circuit with 3 stable states:

- $V_Q = 0$ and $V_{Q-bar} = V_{DD}$,
- $V_Q = V_{DD}$ and $V_{Q-bar} = 0$,
- $V_Q = V_{Q-bar} = V_{inv}$.

The first two states are the logically legal states of the latch while the third state is the illegal *metastable* state. If the circuit is exactly in the metastable state, it will remain there, but if V_Q and V_{Q-bar} are displaced even an ϵ from equality, the voltage difference will be amplified by the feedback and the circuit will eventually settle in one of the logically legal states. As will be shown shortly, this settling may take an arbitrarily long time.

[1]Of course in NMOS latches, the devices must be sized correctly in order to achieve the proper logic voltages but that sizing is really a separate issue.

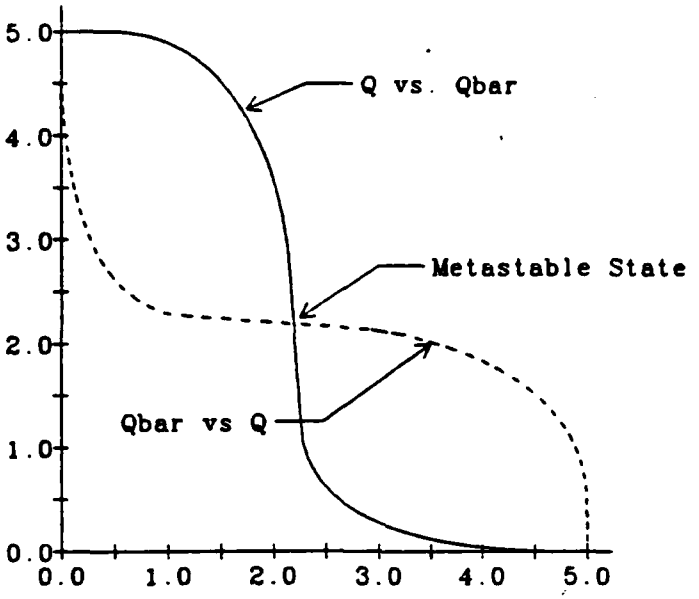


Figure A.2: The Transfer Characteristics of the Bistable Latch

A.2 Predicting the Failure Rate of CMOS Bistable Latches

The analysis of the CMOS bistable latch will consist of two stages which reflect the two different operating modes of the latch. When a transition on the data input begins propagating through the latch, the MOSFETs' operating characteristics vary greatly due to the nonlinear nature of the devices; predicting the behavior the circuitry in this nonlinear operating regime requires fairly sophisticated modeling methods. If a clock edge manages to put the latch into a metastable state, the circuitry will remain in the vicinity of the latch's metastable point for a significant period of time; during this time the circuitry can accurately be described as being linear and time invariant.

The first step of the analysis will use a circuit simulator to model the operation of the latch when the clock and data transition try to force the latch into the metastable state; from these simulations the relationship of the phase margin between the clock and data edges, t_{PM}, to the initial differential voltage latched at the outputs of the bistable latch, $v_d(t_0)$ will be

determined. The second step will be use a small signal model of the latch to study the amplification of $v_d(t_0)$. The results of these two steps can then be combined to predict the probability that latch failures will occur.

A.2.1 Simplifying Assumptions

Definition of Latch Failure

The discussion of noise and phase margins in Section 2.5 defined latch failure as the failure of the outputs voltages, V_Q and V_{Q-bar}, to reach the logic thresholds which can be assumed to be equal to the thresholds of the n- and p-type devices with no loss of generality. This definition of latch failure is more conservative than really necessary and will also cause some problems with the linear analysis of the latch so a slightly less conservative definition will be used in this Appendix. The outputs of the dynamic latches which latch V_Q and V_{Q-bar} on ϕ_2 will be used to determine the failure criteria. Further, the assumption will be made that as the static latch settles out of a metastable state, V_Q and V_{Q-bar} will be changing much slower than the dynamic latches can switch. If $H_D(V_Q)$ is the DC transfer characteristic of the dynamic latches then as the bistable latch is moving towards a stable state, it can safely be assumed that $V_{Data} = H_D(V_{Q-bar})$.

Defining V_M to be the voltage at which $H_D(V_M) = V_{T_n}$, a latch failure can be defined to occur when neither V_Q nor V_{Q-bar} are greater than V_M when the outputs are latched at T_S. This conforms roughly to the definition of latch failure used in [15] and seems to be a fairly reasonable definition. A simulation of the static characteristics of the dynamic latch gives:

$$V_M = 2.85$$

Latched Inputs

The inputs to the DDA synchronizer are completely asynchronous; the inputs to the bistable latches on the other hand are all latched by dynamic latches. Latching the inputs to the bistable latches will limit the types of

transitions the bistable latches see; specifically, the input to the latches will not change for $T_C/2$ seconds before the bistable latches' clock goes LOW. This stability lowers the probability of the latch failing but does not prevent the bistable latches from failing; the input could be held at a voltage which will place the bistable latch in a metastable state. The exact characteristics of the latch are sensitive to the relationship between the clock signals clocking the dynamic and static latches; this adds another dimension to the analysis. The initial analysis of the latch will assume the input to the bistable is not clocked in order to simplify the analysis.

Symmetry

Although the latch in the DDA synchronizer only utilized one output and was therefore very asymetrical , the latch in the analysis will be assumed to be symmetrically designed and loaded. This assumption simplifies the small signal analysis of the latch greatly without sacrificing much insight. Some accuracy is obviously sacrificed but the analysis should give conservative results since the loading on the outputs was taken to be equal to the largest load seen by the DDA synchronizer's latch. The asymmetrical latch can be analyzed and a simple analysis is given in Section A.2.5; in order to achieve results that can be interpreted reasonably, at least as much accuracy was simplified as in the symmetrical analysis.

Noise

The effect of noise on the operation of synchronizers has been studied by several authors[15][8][12]. The types of noise considered, the techniques used and the assumptions the authors made varied somewhat. All of the studies modeled the noise as being purely random with a normal probability distribution, zero mean amplitude and rms values small enough that the linear small signal model was still valid. Noise of this type arises from different sources; the most prevalent sources of noise are the active devices themselves; MOS devices produce a significant amount of thermal and shot

noise both of which produce normally distributed noise currents[28]. There are also other sources of noise, some can be considered small signal while other must be treated as large signal disturbances; only noise which can be reasonably modeled in a manner similar to the thermal and shot noise will impact the considerations of this section. Noise sources which are more deterministic and have larger amplitudes are discussed in Section 2.5.

Two of the studies analyzed the noise effects in a fairly thorough analytical manner[15][8]; the other studies used more empirical methods including graphical and textual arguments. The conclusion all of the studies have drawn is that noise does not effect the probability that a synchronizer failure will occur. The conclusions can be summed up by the argument that if the noise amplitudes are truly random, then when the synchronizer is trying to escape from the metastable region, the probability that noise will force the latch back into the metastable region is equal to the probability that noise will force the latch out of the metastable region.

A.2.2 Simulation of the CMOS Bistable Latch

Most of the past studies have used some type of linear approximation to predict $v_d(t_0)$; while the inaccuracies introduced by the approximations are not significant enough to invalidate the results, no great insight into the problem is lost by using a simulator to make more accurate predictions.

The simulator used for these simulations, SPICE2G.5, effectively models many of the second order effects which are important in accurately predicting the performance of MOS circuits. The models used by SPICE are fairly complicated and their accuracy depends heavily on how well the parameter values used match the specific process.

Many of the parameters of the models are empirical in nature and a common procedure for determining the parameters to use is to make a series of measurements on test structures that have been fabricated using the target process and then perform a global curve fitting operation to match SPICE predictions to the observed behavior. This procedure results in the best

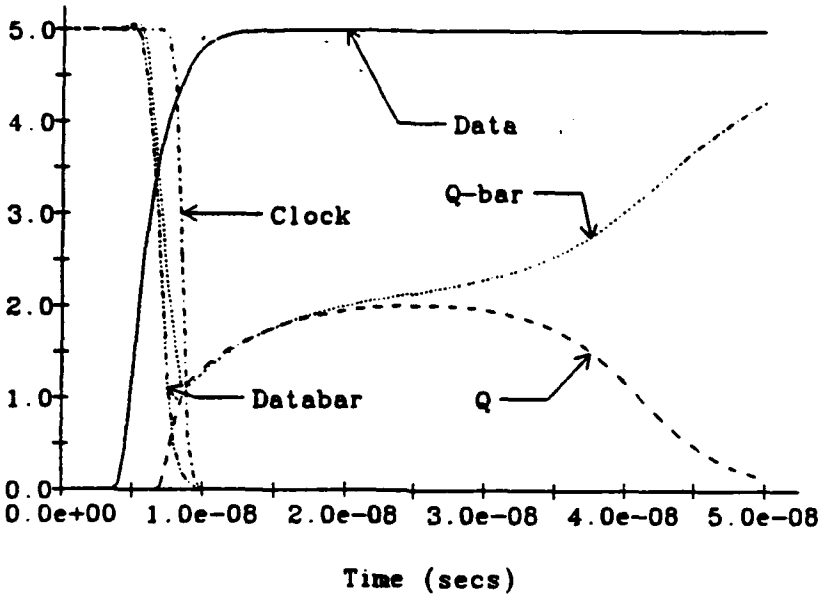


Figure A.3: SPICE2.G5 Simulation of the CMOS Bistable Latch

set of parameter values but the actual values may not have any intuitively obvious relationship to the processing details. The SPICE models used for the simulations in this thesis were extracted in this manner by the MOSIS fabrication service[6].

Figure A.3 shows the simulation of the bistable latch for one particular phase relationship between the clock and data edges. In the simulation shown, the rising input causes the Q-bar output and the Data-bar input to begin falling. The Data-bar input lags the Data input by an inverter delay and therefore the Q output does not start rising until the Q-bar output has fallen to almost V_{T_n}. This behavior is slightly different from the behavior caused by falling input transitions which violate the setup requirements of the latch; Figure 2.3(b) shows how a falling transition causes the outputs to be latched much nearer the metastable point of the latch. The analysis in this Appendix does not depend on what type of transition causes the metastable states to appear; all discussions will use the rising transition response shown in Figure A.3 as a reference example.

In this analysis, all time measurements will be related to the point at

which the clock input is equal to V_{T_n}; this time will be referred to as t_0. The phase margin of the input signal will still be determined by the $V_{DD}/2$ points of the Data and Clock signals. The change in time reference for the latch responses should not affect the outcome of the analysis.

After t_0, the two cross-coupled inverters are isolated by the high-impedance outputs of the tristate latches[2]. In the absence of noise, the latch's response and final state are completely dependent on $V_Q(t_0)$ and $V_{Q\text{-}bar}(t_0)$.

SInce $V_Q(t_0)$ and $V_{Q\text{-}bar}(t_0)$ are both close to V_{T_n} the n-type transistors are both almost cutoff. As a result, Q and Q-bar begin to be pulled HIGH, at identical speeds, by the p-type pullups.

As V_Q and $V_{Q\text{-}bar}$ increase, the n-type pulldowns gradually turn on, initiating the regenerative action of the cross-coupled inverters. The maximum loop gain is obtained at the metastable point of the latch but the gain will be close to maximum as long the operating point is within a threshold or so of the V_{inv}; the critical factor is that the all of the transistors remain saturated and this will be the case as long as $|V_Q - V_{Q\text{-}bar}| < V_T$.

When V_Q and $V_{Q\text{-}bar}$ both rise to the vicinity of the V_{inv} point of the inverters, the circuit is very nearly balanced at its metastable point. Unless V_Q and $V_{Q\text{-}bar}$ are exactly equal, the feedback present in the latch will amplify the differential voltage, forcing the latch to stable, logically legal states. The plot seems to indicate that the amplification of V_Q and $V_{Q\text{-}bar}$ to stable states is exponential in nature; this fact will be confirmed by the linear, small signal analysis in the next section.

For the particular arrangement of the clock and data signals simulated, the latch settles to the state $V_Q = 0$ and $V_{Q\text{-}bar} = V_{dd}$ which happens to be the state of the latch before the transitions occurred. Had the data transition occurred 50 picoseconds earlier, this outcome would have been reversed. By performing a series of simulations in which the data transition point varied by only a few picoseconds, the characteristics of the latch in the critical switching region can be determined. Simulations were performed with a

[2]Assuming clock-bar rises above V_{T_p} at t_0 also.

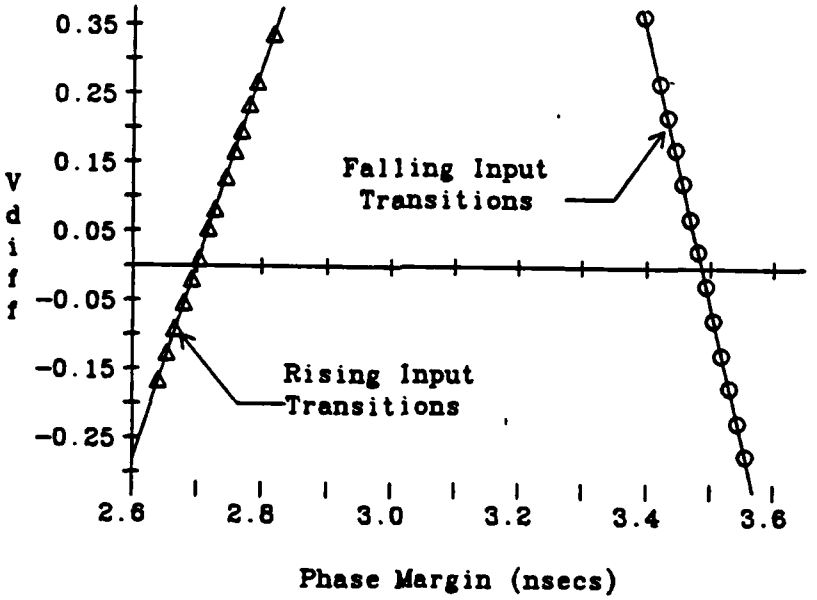


Figure A.4: Plot of $v_d(t_0)$ vs. Phase Margin

range of phase margins and both falling and rising Data signals; Figure A.4 plots the initial differential voltage, $v_d(t_0) = V_Q(t_0) - V_{Q-bar}(t_0)$ versus the phase margin, t_{PM}.

For initial differential voltages of a few hundred millivolts or less, the dependence of $v_d(t_0)$ on t_{PM}, $v_{d0}(t_{PM})$, can be approximated by a linear relationship having a slope of S and a zero point of $t_{PM} = t_M$:

$$v_{d0}(t_{PM}) = S(t_{PM} - t_M) \qquad (A.1)$$

Falling and rising inputs will produce linear relationships with different slopes, S_F and S_R, and different zero points, t_{M_F} and t_{M_R}. The simulations indicated that

$$S_F = -3.9 \text{ V/ns} \qquad t_{M_F} = 3.5 \text{ ns}$$

and

$$S_R = 2.8 \text{ V/ns} \qquad t_{M_R} = 2.7 \text{ ns}$$

These relationships between $v_d(t_0)$ and T_{PM} make it possible to develop a simple expression for $P_v(v_d(t_0) < v_x)$, the probability that $v_d(t_0)$ will be less than some value, v_x. Due to the asynchronous nature of the input the

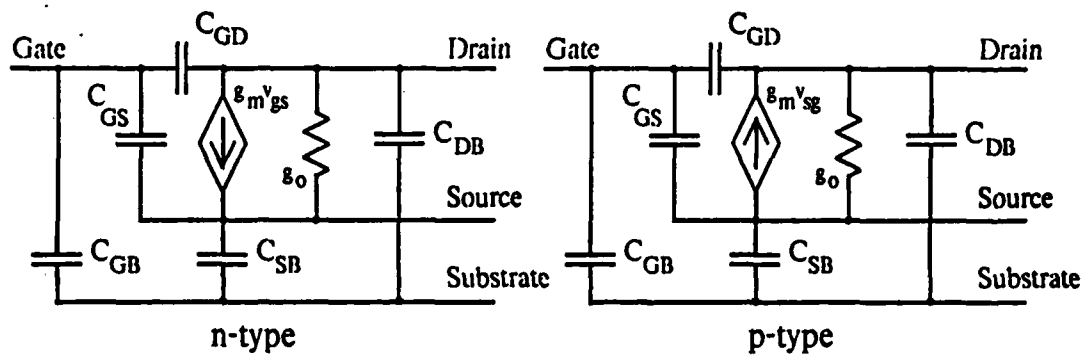


Figure A.5: Small Signal Models for n- and p-type MOSFETs

data transitions can be assumed to have an equal probability of occurring at any time, i.e. the probability density of both rising and falling transitions will be $p_D = 1/T_D$ where T_D is the average period between falling or rising transitions and $T_D \geq 2T_C$. The probability $P_v(v_d(t_0) < v_M)$ can be found by adding the probabilities arising from falling and rising transitions which can in turn be found from the slopes S_F and S_R:

$$P_v(v_d(t_0) < v_x) = p_D\left(\left(\frac{2v_x}{S_F}\right) + \left(\frac{2v_x}{S_R}\right)\right) = \frac{2v_x}{T_D}\left(\frac{1}{S_F} + \frac{1}{S_R}\right). \qquad (A.2)$$

A.2.3 Linear Analysis

When the bistable latch is placed very close to its metastable state, as was the case in the simulation illustrated in Figure A.3, the differential voltage may easily remain less the a threshold for a significant period of time. During this period, all 4 devices which constitute the cross-coupled inverters are operating in their saturation regimes. In this region the variations in the device characteristics will be small enough that modeling the devices by their linearized small signal models is justified.

Linearized Small Signal Models

Figure A.5 illustrates the small signal models used for the n- and p-type transistors. The model consists of a voltage controlled current source,

an output impedance and 5 capacitances. The transconductances, g_{m_n} and g_{m_p}, and output impedances, g_{o_n} and g_{o_p}, can be found by linearizing the relationship between drain current and the node voltages around the operating point:

$$g_{m_n} = \frac{\partial I_{DS_n}}{\partial V_{GS_n}} = \frac{\partial I_{DS_n}}{\partial V_G} \tag{A.3}$$

$$g_{m_p} = \frac{\partial I_{SD_p}}{\partial V_{SG_p}} = -\frac{\partial I_{SD_p}}{\partial V_G} \tag{A.4}$$

$$g_{o_n} = \frac{\partial I_{DS_n}}{\partial V_{DS_n}} \tag{A.5}$$

$$g_{o_p} = \frac{\partial I_{SD_p}}{\partial V_{SG_p}} \tag{A.6}$$

The capacitance model of the MOSFET is simplified slightly when the device is saturated. Both the channel-to-substrate and the gate-to-drain capacitances can be neglected and, in saturation, the gate-to-source capacitance is $C_{GS} = \frac{2}{3}WLC_{OX}$. The source and drain capacitances, C_{SB} and C_{DB}, are the nonlinear capacitances associated with the diffusion area of the source and drain; these can be determined by extracting the diffusion areas and perimeters from the layout. The other parasitics and load capacitances can be determined by extracting the layout in a similar manner. Due to the cross-coupling nature of the circuit, there may also be a capacitance coupling the Q and $Q-bar$ nodes together.

Figure A.6 shows the resulting small signal model for the cross-coupled inverters. The symmetrical nature of the latch's devices and loads results in a symmetrical small signal model also. The capacitances shown represent the combination of all of the device, load and parasitic capacitances.

KCL equations can be written for the 2 nodes in the small signal model in terms of v_Q and v_{Q-bar}:

$$\dot{v}_Q(C_L + C_C) - \dot{v}_{Q-bar}C_C + v_Q(g_{o_p} + g_{o_n}) + v_{Q-bar}(g_{m_p} + g_{m_n}) = 0 \tag{A.7}$$

$$\dot{v}_{Q-bar}(C_L + C_C) - \dot{v}_Q C_C + v_{Q-bar}(g_{o_p} + g_{o_n}) + v_Q(g_{m_p} + g_{m_n}) = 0 \tag{A.8}$$

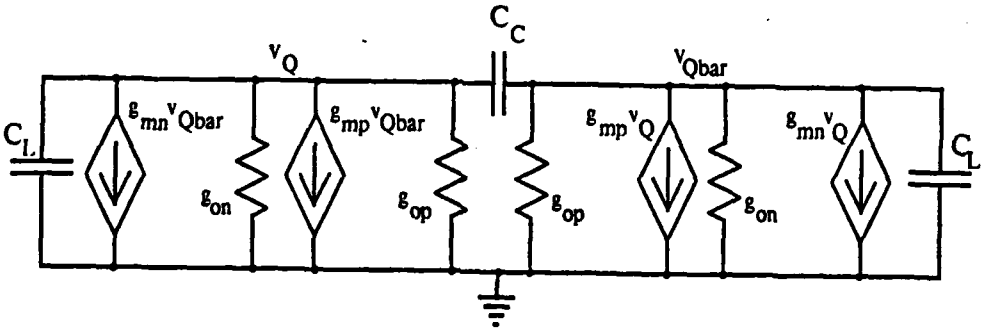


Figure A.6: Small Signal Model for the CMOS Bistable Latch

Equations A.7 and A.8 can be combined to determine $v_c(t) = v_Q(t) + v_{Q-bar}(t)$ and $v_d(t) = v_Q(t) - v_{Q-bar}(t)$, the common-mode and differential signals seen by the latch.

$$\dot{v}_c C_L + v_c(g_{o_p} + g_{o_n} + g_{m_n} + g_{m_p}) = 0 \qquad (A.9)$$

$$\dot{v}_d(C_L + 2C_C) + v_d(g_{o_p} + g_{o_n} - g_{m_n} - g_{m_p}) = 0. \qquad (A.10)$$

Equations A.9 and A.10 produce the common-mode and differential responses of the metastable latch:

$$\begin{aligned} v_c(t) &= v_c(t_0)\exp\left[\frac{-(g_{o_n}+g_{o_p}+g_{m_n}+g_{m_p})}{C_L}(t-t_0)\right] \\ &= v_c(t_0)exp\left[-(t-t_0)\frac{1}{\tau_c}\right] \end{aligned} \qquad (A.11)$$

$$v_d(t) = v_d(t_0)\exp\left[\frac{g_{m_n}+g_{m_p}-g_{o_n}-g_{o_p}}{C_L+2C_C}(t-t_0)\right] = v_c(t_0)\left[(t-t_0)\frac{1}{\tau_d}\right]. \qquad (A.12)$$

These solutions predict that the common-mode signal will decay exponentially while the differential signal will grow exponentially[3]; these predictions fit well with the behavior illustrated in Figure A.3.

Estimation of the Parameters

In order to predict the latch's behavior quantitatively as well as qualitatively, the parameters of the small signal model must be determined. The

[3]Provided $g_{m_n} + g_{m_p} > g_{o_n} + g_{o_p}$, e.g. the inverter gain must be greater than 1; this is already the case for restoring digital logic circuits[12] so this restriction is not important.

capacitances can be found by an extraction of the layout of the latch; since the actual latch used was asymmetrical, the largest loading predicted by the circuit extractor will be used for C_L. The worst case load capacitance comes out to be $C_L = 560 \times 10^{-15}$F; but there is no coupling capacitance due to the way in which the cross-coupled inverters were layed out and to the assumption that $C_{GD} = 0$ for the saturated transistors.

Rather than try and perform the linearization of the current relationships by hand the g_m's and g_o's were obtained from the operating point information provided by SPICE; the brief discussion of some of the second order effects present in MOSFETs given in the next Section should make it apparent why SPICE was used for this purpose. When the inverters were placed at the metastable point of the latch, the small signal parameters were found to be:

$$\begin{aligned} g_{m_n} &= 6.74 \times 10^{-5}\ A/V \\ g_{m_p} &= 4.13 \times 10^{-5}\ A/V \\ g_{o_n} &= 2.45 \times 10^{-6} \\ g_{o_p} &= 5.43 \times 10^{-6}\ . \end{aligned}$$

Given these parameters, the time constants of the common-mode and differential signals can be determined:

$$\tau_d = 5.6 \text{ nsec}$$

$$\tau_c = 4.8 \text{ nsec.}$$

These time constants are somewhat larger than was really expected. The type of bistable latch chosen required the devices in the inverters be small in order for the dynamic latches to be able to overpower the inverters. The small inverter devices coupled with a rather sizable load of over $\frac{1}{2}$ picoFarad resulted in a large RC time constant. Even so the settling time required to achieve the extremely low failure probability was not long enough to have a serious impact on the performance of any the system which uses the synchronizer. The shorter settling time produced by an improved bistable latch design could help reduce the area requirements of the synchronizer by increasing the number of synchronizers that could share a single set of control logic.

Second Order Effects in MOSFETS

The classical model of a MOSFET operating with $(V_{GS} - V_T) < V_{DS}$ predicts a quadratic dependence of I_{DS} on $(V_{GS} - V_T)$ but no dependence on the voltage at the drain. There are several second order effects which have noticeable effects on the current-voltage relationships, especially for devices with short channels lengths. Three of these second order effects will be discussed here in order to give a feeling for how these effects impact the analysis of the static latch: drain induced barrier lowering, carrier velocity saturation and channel length modulation. A thorough discussion of these and other characteristics is given Chapter 2 of [12].

In short channel devices, the drain region acts like a second gate with a much thicker oxide; as the drain voltage rises above the substrate potential (or drops below it for p-type devices) the voltage on the drain tends to induce charges into the channel in the same manner the gate does, effectively lowering the threshold of the device. This characteristic is modeled by introducing a feedback term reflect the drain voltage dependence of the device thresholds:

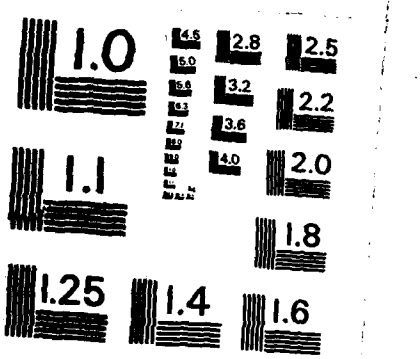
$$V'_{T_n} = V_{T_n} - \sigma_n V_{DS} = V_{T_n} - \sigma_n V_D$$

$$V'_{T_p} = V_{T_p} - \sigma_p V_{SD} = V_{T_p} - \sigma_p (V_{DD} - V_D).$$

Accounting for velocity saturation effects is more involved. The simple model for the channel current models the velocity of the carriers in the channel as being directly proportional to the electric fields present in the channel region. This relationship does hold for moderate strength fields; at higher field strengths the velocities reach a maximum velocity, ν_{MAX}, and any further increases in the electric field do not increase the speed of the carriers. In the classical model the electric field in a saturated device was limited to $V_{DSAT} = (V_{GS} - V_T)/L$; this dependence of the electric field on $(V_{GS} - V_T)$ coupled with a similar dependence of the charge in the channel produced the quadratic dependence of the current on $(V_{GS} - V_T)$. Once the carriers our saturated, the drain current will only be linearly dependent on $(V_{GS} - V_T)$.

AD-A166 846 A HIGH-SPEED ASYNCHRONOUS COMMUNICATION TECHNIQUE FOR MOS (METAL-OXIDE-SE (U) MASSACHUSETTS INST OF TECH CAMBRIDGE DEPT OF ELECTRICAL ENGIN P D BASSETT

UNCLASSIFIED DEC 85 VLSI-MEMO-85-283 N00014-80-C-0622 F/G 20/12 NL



At low field levels, the mobility of electrons is much higher than that of holes and as a result the electrons travel at higher velocities; the saturation velocities of electrons and holes are much closer together than the low-field mobilities so the performance of p-type devices is much closer to that of n-type devices when velocity saturation is a major factor.

A third effect which is important in saturated MOSFETs is the dependence of the effective channel length on the drain voltage. If the potential in the channel of a saturated MOSFET, ϕ_S, were measured starting at the source and progressing towards the drain, the potential would increase continuously until $\phi_S = V_{DSAT}$. At the point in the channel at which $\phi_S = V_{DSAT}$, the charge density would have decreased to the point that the channel was *pinched off*; any further increases in ϕ_S would cutoff the current flow. A depletion region forms between the pinch off point and the drain of the device; the excess drain voltage, $V_{DS} - V_{DSAT}$, appears as the voltage across the depletion region. As V_{DS} increases, the voltage across the depletion region will also increase causing the width of the depletion region to increase also. Any increase in the width of the depletion region shortens the effective length of the device resulting in an increase in the current.

This channel length modulation effect is modeled by substituting an effective length for the actual length:

$$L' \doteq L - \Lambda.$$

There are a number of ways to model Λ with both empirical and physical models of varying complexity; one complicating factor is that Λ is dependent on L' and L' is dependent on Λ.

A.2.4 Failure Probability

This section will combine the results generated in the previous section to obtain an equation for predicting the probability of failure of the bistable latch used in the DDA synchronizer implementation. A synchronization error was defined previously as occurring when neither V_Q nor $V_{Q\text{-}bar}$ are

greater than V_M by the end of the settling time. Given the assumption of a balanced response, a failure can be defined as occurring when $v_d(T_S) < v_M$ where $v_M = 2(V_M - V_{inv})$. The probability of an error occurring is given by the integral of the probability density function for the differential voltage at the end of the settling time over the range of metastable voltages:

$$P_M(T_S) = \int_{-v_M}^{v_M} p_v(v_d, T_S) dv_d. \tag{A.13}$$

The response of the latch to an initial differential voltage given by Equation A.12 indicate that in order for a failure to occur, the initial differential voltage would have to be

$$v_d(t_0) < v_M \exp\left[-\frac{T_S}{\tau_d}\right] \tag{A.14}$$

in order for

$$v_d(T_S) < v_M.$$

The probability density at the end of the settling time is therefore directly dependent on the probability density at t_0:

$$p_v(v_d, T_S) = p_v(v_d \exp\left[-\frac{T_S}{\tau_d}\right], t_0). \tag{A.15}$$

The results of Section A.2.2 indicate that the probability density of $v_d(t_0)$ for small differential voltages is linear and can be estimated as:

$$p_v(v_d, t_0) = \frac{1}{T_D}\left(\frac{1}{|S_f|} + \frac{1}{|S_r|}\right). \tag{A.16}$$

Combining Equations A.13, A.15 and A.16 yields the probability of the failure in terms of the settling time allowed, T_S, or the settling time required to achieve a given P_M.

$$P_M(T_S) = \frac{2v_M}{T_D}\left(\frac{1}{|S_f|} + \frac{1}{|S_r|}\right) \exp\left[-\frac{T_S}{\tau_d}\right] \tag{A.17}$$

$$T_S = -\tau_d \ln\left(\frac{P_M T_D}{2v_M\left(\frac{1}{|S_f|} + \frac{1}{|S_r|}\right)}\right) \tag{A.18}$$

The specification for the maximum failure probability in the high speed multiprocessor was $P_F < 10^{-20}$, this was assuming a 100MHz clock which is about twice as fast as the 3 micron chip can operate but a factor of 2 in the P_M specification is negligible. Assuming $T_D = T_C = 20 \times 10^{-9}$ seconds:

$$T_S = -5.6 \times 10^{-9} \ln \left(\frac{10^{-20} \times 20 \times 10^{-9}}{2 \times 0.68 \left(\frac{1}{3.9 \times 10^9} + \frac{1}{2.8 \times 10^9} \right)} \right) = 240 \text{ nsec.}$$

In order to meet the failure probability specification, the dynamic latches would have to given 240 nanoseconds or 12 clock cycles to settle.

A.2.5 Solutions for Non-Symmetrical Loads

Solutions developed in Section A.2.3 gave expressions for the common-mode and differential gain for a symmetrically loaded latch. In the configuration implemented in the DDA synchronizer, only the Q-bar output was utilized resulting a asymmetric loading on the latch outputs. The results must be extended in order to accurately predict the effect of the asymmetry on the latch's performance.

In order to simplify matters, two assumptions will be made. First, the coupling capacitance, C_C, will be ignored[4]. Secondly, the effect of drain induced barrier lowering on the device thresholds will be ignored. These two assumptions greatly simplify the problem, hopefully without sacrificing too much in accuracy.

The small signal model resulting from these simplifications is shown in Figure A.7. The dynamic performance of this circuit can be described by the following equation:

$$\dot{\mathbf{v}}_= \begin{bmatrix} \dot{v}_Q \\ \dot{v}_{Q-bar} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{C_1}(g_{m_n} + g_{m_p}) \\ \frac{1}{C_2}(g_{m_n} + g_{m_p}) & 0 \end{bmatrix} \begin{bmatrix} v_Q \\ v_{Q-bar} \end{bmatrix}. \quad (A.19)$$

The solutions to this equation will be of the form

$$v_Q(t) = k_1 \exp[\lambda_1 t] + k_2 \exp[\lambda_2 t] \quad (A.20)$$

[4]The small signal gain could be used to divided this Miller capacitance into its *equivalent* capacitances on the inputs and outputs of the inverters.

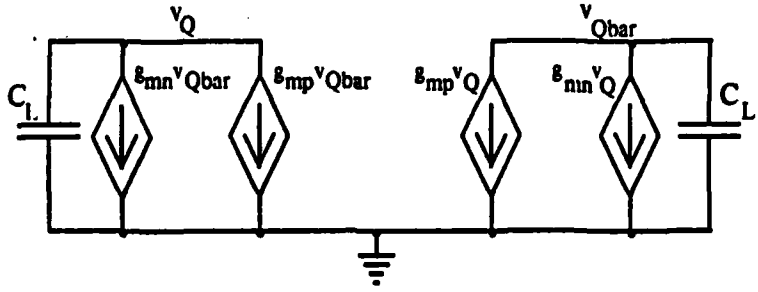


Figure A.7: The Simplified Small Signal Model of the DDA Synchronizer's Bistable Latch

and

$$v_{Q-bar}(t) = k_3 \exp[\lambda_1 t] + k_4 \exp[\lambda_2 t]. \qquad (A.21)$$

In this case it is straightforward to show:

$$\lambda_1 = -\lambda_2 = \frac{1}{\sqrt{C_1 C_2}}(g_{m_n} + g_{m_p}),$$

$$k_1 = -\sqrt{\frac{C_2}{C_1}}k_3 = \frac{1}{2}\left[v_1(t_0) - \sqrt{\frac{C_2}{C_1}}v_2(t_0)\right],$$

$$k_2 = \sqrt{\frac{C_2}{C_1}}k_4 = \frac{1}{2}\left[v_1(t_0) + \sqrt{\frac{C_2}{C_1}}v_2(t_0)\right].$$

Substituting into the expressions for $v_Q(t)$ and $v_{Q-bar}(t)$ produces:

$$\begin{aligned} v_Q(t) \;=\; & \tfrac{1}{2}\Big[\Big(v_Q(t_0) - \sqrt{\tfrac{C_2}{C_1}}v_{Q-bar}(t_0)\Big)\exp[\lambda_1 t] \\ & + \Big(\big(v_Q(t_0) + \sqrt{\tfrac{C_2}{C_1}}v_{Q-bar}(t_0)\big)\exp[-\lambda_1 t]\Big] \end{aligned} \qquad (A.22)$$

$$\begin{aligned} v_{Q-bar}(t) \;=\; & \tfrac{1}{2}\Big[-\Big(v_Q(t_0) - \sqrt{\tfrac{C_2}{C_1}}v_{Q-bar}(t_0)\Big)\exp[\lambda_1 t] \\ & + \Big(\big(v_Q(t_0) + \sqrt{\tfrac{C_2}{C_1}}v_{Q-bar}(t_0)\big)\exp[-\lambda_1 t]\Big] \end{aligned} \qquad (A.23)$$

The behavior predicted by Equations A.22 and A.23 could almost have been predicted simply by extrapolation from the responses predicted for the symmetrical load situation. The responses of v_Q and v_{Q-bar} each consists of a common-mode response and a differential mode response as evidenced by

the presence of $(v_Q(t_0) - v_{Q\text{-}bar}(t_0))$ and $(v_Q(t_0) + v_{Q\text{-}bar}(t_0))$ factors in both responses. Further, the differential portion is growing exponentially while the common-mode portion is decaying exponentially. The nonsymmetry of the loading is reflected by the $\sqrt{\frac{C_2}{C_1}}$ scaling factor which will cause the response of the more heavily loaded node to be slower than the other output.

Bibliography

[1] Bell, A. G. and Borriello, G., "A Single Chip NMOS Ethernet Controller," *Proceeding of the 1983 IEEE International Solid-State Circuits Conference*, pp. 70-71.

[2] Burns, J. R., "Switching Response of Complementary Symmetry MOS Transistor Logic Circuits," *RCA Review*, pp. 627-661, 1964.

[3] Catt, Ivor, "Time Loss Through Gating of Asynchronous Logic Signal Pulses," *IEEE Trans. on Electronic Computers*, Vol. EC-15, No. 1, February 1966, pp 108-111.

[4] Chaney and Molner, "Anomalous Behavior of Synchronizer and Arbiter Circuits," *IEEE Trans. on Computers*, Vol. C-22, No. 4, April 1973, pp 421-422.

[5] Chaney, Thomas J. and Fred U. Rosenberg, "Characterization and Scaling of MOS Flip Flop Performance in Synchronizer Applications," *Proc., Caltech Conference on Very Large Scale Integration 1979*, January 1979, pp 357-374.

[6] Cohen, D., G. Lewicki, "MOSIS - The ARPA Silicon Broker," *Proceedings from the Second Caltech Conference on VLSI*, California Institute of Technology, Pasadena, CA, January 1981, pp. 29-44.

[7] Couranz, G. R., "An Analysis of Binary Circuits Under Marginal Triggering Conditions," Techinal Memorandum #15, Computer Systems Lab, Washington University, St. Louis, Nov 1969.

[8] Couranz, George R. and Donald F. Wann, "Theoretical and Experimental Behavior of Synchronizers in the Metastable Region," *IEEE Trans. on Computers*, Vol C-24, June 1975, pp. 604-616.

[9] Corsini, P., "Self-Synchronizing Asynchronous Arbiter," *Digital Processes*, Vol. 1, Spring 1975, pp 67-73.

[10] Gardner, F. M., **Phaselock Techniques**, John, Wiley & Sons, New York, New York, 1979.

[11] Gardner, F. M., "Charge-Pump Phase-Lock Loops," *IEEE Trans. on Communications*, Vol. COM-28, No. 11, Nov. 1980, pp. 1849-58.

[12] Glasser, L. A. and Dobberpuhl, D., **The Design and Analysis of VLSI Circuits**, Addison-Wesley, 1985.

[13] Glasser, L. A., "The Analog Behavior of Digital Integrated Circuits," VLSI Memo 80-36, Dept. of Electrical Engineering and Computer Science, M.I.T., Cambridge, MA, December 1980.

[14] Glasser, L. A., "Delay, Noise Margin and Gain in Digital Circuits," to be published in *Proceedings of the Chapel Hill Conference on VLSI*, May 1985.

[15] Hohl, J. H., Larsen, W. R., and Schooley, L. C., "Prediction of Error Probabilities for Integrated Digital Synchronizers," *IEEE Journal of Solid-State Circuits*, Vol. SC-19, No. 2, April 1982, pp. 236-244.

[18] Kanuma, A., "CMOS Circuit Optimization," *Solid-State Electronics*, Vol. 26, No. 1, 1983, pp. 47-58.

[19] Kinniment, D. J. and J. V. Woods, "Synchronization and Arbitration Circuits in Digital Systems," *Proc. of the IEE (England)*, Vol. 123, No. 10, October 1976, pp 961-66.

[20] Littlefield, W. M. and T. J. Chaney, "The Glitch Phenomenon," Tech. Memorandum #10, Computer Systems Laboratory, Washington University, St. Louis, Dec 1966.

[21] Marino, Leonard R., "The Effect of Asynchronous Inputs on Sequential Network Reliability," *IEEE Trans. on Computers*, Vol. C-26, No. 11, November 1977, pp 1082-90.

[22] Mars, P., "Study of the Probabilistic Behavior of Regenerative Switching Circuits," *Proc. of the IEE (England)*, vol 115, May 1968, pp 662-668.

[23] McNamara, J. E., **Technical Aspects of Data Communications**, Digital Press, Digital Equipment Corporation, Bedford, MA 1982.

[24] Mead, C. A. and Conway L., **Introduction to VLSI Systems**, Addison-Wesley, Reading, MA, 1980.

[25] Nagel. L. W., "SPICE2, A Computer Program to Simulate Semiconductor Circuits," ERL-M520, Electronics Research Laboratory, U.C. Berkeley, CA, May 1975.

[26] Pechoucek, M., "Anomalous Response Times of Input Synchronizers," *IEEE Trans. on Computers*, Vol. C-25, No. 2, Febrary 1976, pp 133-139.

[27] Plummer, William W., "Asynchronous Arbiters," *IEEE Trans. on Computers*, Vol. C-21, No. 1, January 1972, pp 37-42.

[28] Papoulis, A., *Probability, Random Variables and Stochastic Processes*, McGraw-Hill: New York, 1965.

[29] Rosenberg, F. U., "Control of Concurrent Operations in Asynchronous Digital Process," Technical Memorandum #14, Computer Systems Lab, Washington University, St. Louis.

[30] Seitz, Charles L., "System Timing," *Chapter 7, Introduction to VLSI Systems,* Carver Mead and Lynn Conway, authors, Addison-Wesley, Reading, MA, 1980.

[31] Seitz, Charles L., "Self-Timed VLSI Systems," *Proc., Caltech Conference on Very Large Scale Integration 1979,* January 1979, pp. 345-355.

[32] Stucki, M. J. and J. R. Cox, Jr., "Synchronization Strategies," *Proc., Caltech Conference on Very Large Scale Integration 1979,* January 1979, pp. 375-393.

[33] Sutherland, Ivan E., "A Specific Arbiter Design, ARBIT2," Computer Science Department, Caltech, Display File No. 226, Sep 1976.

[34] Taft, J. D., "A Calibrated Digital Delay Line Implemented in VLSI," Bachelor's Thesis, MIT, January 1983.

[35] —, The Synchronization Problem, "Technical Report 47," Washington University Computer Systems Lab, February 1974.

[36] Veendrick, H. J. M.,, 'Behavior of Flip-Flops Used as Synchronizers and Prediction of Their Failure Rates," *IEEE Journal of Solid-State Circuits,* Vol. SC-15, No. 2, April 1980, pp 169-176.

END

DTIC

5-86